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

LNAI 7523

Machine Learning and Knowledge Discovery in Databases

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

Lecture Notes in Artificial Intelligence 7523

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 I

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

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-33459-7 e-ISBN 978-3-642-33460-3
DOI 10.1007/978-3-642-33460-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012946760

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 I

Invited Talks

Machine Learning for Robotics 1
Pieter Abbeel

Declarative Modeling for Machine Learning and Data Mining 2
Luc De Raedt

Machine Learning Methods for Music Discovery and Recommendation 4
Douglas Eck

Solving Problems with Visual Analytics: Challenges and Applications ... 5
Daniel Keim

Analyzing Text and Social Network Data with Probabilistic Models 7
Padhraic Smyth

Association Rules and Frequent Patterns

Discovering Descriptive Tile Trees: By Mining Optimal Geometric Subtiles 9
Nikolaj Tatti and Jilles Vreeken

Efficient Discovery of Association Rules and Frequent Itemsets through Sampling with Tight Performance Guarantees 25
Matteo Riondato and Eli Upfal

Smoothing Categorical Data 42
Arno Siebes and René Kersten

Bayesian Learning and Graphical Models

An Experimental Comparison of Hybrid Algorithms for Bayesian Network Structure Learning 58
Maxime Gasse, Alex Aussem, and Haytham Elghazel

Bayesian Network Classifiers with Reduced Precision Parameters 74
Sebastian Tschiatschek, Peter Reinprecht, Manfred Mücke, and Franz Pernkopf

Combining Subjective Probabilities and Data in Training Markov Logic Networks 90
Tivadar Pápai, Shalini Ghosh, and Henry Kautz

Score-Based Bayesian Skill Learning 106
Shengbo Guo, Scott Sanner, Thore Graepel, and Wray Buntine

Classification

A Note on Extending Generalization Bounds for Binary Large-Margin Classifiers to Multiple Classes 122
Ürün Dogan, Tobias Glasmachers, and Christian Igel

Extension of the Rocchio Classification Method to Multi-modal Categorization of Documents in Social Media 130
Amin Mantrach and Jean-Michel Renders

Label-Noise Robust Logistic Regression and Its Applications 143
Jakramate Bootkrajang and Ata Kabán

Sentiment Classification with Supervised Sequence Embedding 159
Dmitriy Bespalov, Yanjun Qi, Bing Bai, and Ali Shokoufandeh

The Bitvector Machine: A Fast and Robust Machine Learning Algorithm for Non-linear Problems 175
Stefan Edelkamp and Martin Stommel

Dimensionality Reduction, Feature Selection and Extraction

Embedding Monte Carlo Search of Features in Tree-Based Ensemble Methods 191
Francis Maes, Pierre Geurts, and Louis Wehenkel

Hypergraph Spectra for Semi-supervised Feature Selection 207
Zhihong Zhang, Edwin R. Hancock, and Xiao Bai

Learning Neighborhoods for Metric Learning 223
Jun Wang, Adam Woznica, and Alexandros Kalousis

Massively Parallel Feature Selection: An Approach Based on Variance Preservation 237
Zheng Zhao, James Cox, David Duling, and Warren Sarle

PCA, Eigenvector Localization and Clustering for Side-Channel Attacks on Cryptographic Hardware Devices 253
Dimitrios Mavroeidis, Lejla Batina, Twan van Laarhoven, and Elena Marchiori

Distance-Based Methods and Kernels

Classifying Stem Cell Differentiation Images by Information Distance ... 269
Xianglilan Zhang, Hongnan Wang, Tony J. Collins, Zhigang Luo, and Ming Li

Distance Metric Learning Revisited 283
Qiong Cao, Yiming Ying, and Peng Li

Geodesic Analysis on the Gaussian RKHS Hypersphere 299
Nicolas Courty, Thomas Burger, and Pierre-François Marteau

Ensemble Methods

Boosting Nearest Neighbors for the Efficient Estimation of Posteriors ... 314
Roberto D'Ambrosio, Richard Nock, Wafa Bel Haj Ali, Frank Nielsen, and Michel Barlaud

Diversity Regularized Ensemble Pruning 330
Nan Li, Yang Yu, and Zhi-Hua Zhou

Ensembles on Random Patches 346
Gilles Louppe and Pierre Geurts

Graph and Tree Mining

An Efficiently Computable Support Measure for Frequent Subgraph Pattern Mining 362
Yuyi Wang and Jan Ramon

Efficient Graph Kernels by Randomization 378
Marion Neumann, Novi Patricia, Roman Garnett, and Kristian Kersting

Graph Mining for Object Tracking in Videos 394
Fabien Diot, Elisa Fromont, Baptiste Jeudy, Emmanuel Marilly, and Olivier Martinot

Hypergraph Learning with Hyperedge Expansion..... 410
Li Pu and Boi Faltings

Nearly Exact Mining of Frequent Trees in Large Networks..... 426
Ashraf M. Kibriya and Jan Ramon

Reachability Analysis and Modeling of Dynamic Event Networks..... 442
Kathy Macropol and Ambuj Singh

Large-Scale, Distributed and Parallel Mining and Learning

CC-MR – Finding Connected Components in Huge Graphs with MapReduce 458
Thomas Seidl, Brigitte Boden, and Sergej Fries

Fast Near Neighbor Search in High-Dimensional Binary Data 474
Anshumali Shrivastava and Ping Li

Fully Sparse Topic Models 490
Khoat Than and Tu Bao Ho

Learning Compact Class Codes for Fast Inference in Large Multi Class Classification 506
M. Cissé, T. Artières, and Patrick Gallinari

ParCube: Sparse Parallelizable Tensor Decompositions 521
Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos

Stochastic Coordinate Descent Methods for Regularized Smooth and Nonsmooth Losses 537
Qing Tao, Kang Kong, Dejun Chu, and Gaowei Wu

Sublinear Algorithms for Penalized Logistic Regression in Massive Datasets 553
Haoruo Peng, Zhengyu Wang, Edward Y. Chang, Shuchang Zhou, and Zhihua Zhang

Multi-Relational Mining and Learning

Author Name Disambiguation Using a New Categorical Distribution Similarity 569
Shaohua Li, Gao Cong, and Chunyan Miao

Lifted Online Training of Relational Models with Stochastic Gradient Methods 585
Babak Ahmadi, Kristian Kersting, and Sriraam Natarajan

Scalable Relation Prediction Exploiting Both Intrarelational Correlation and Contextual Information 601
Xueyan Jiang, Volker Tresp, Yi Huang, Maximilian Nickel, and Hans-Peter Kriegel

Relational Differential Prediction 617
Houssam Nassif, Vítor Santos Costa, Elizabeth S. Burnside, and David Page

Multi-Task Learning

Efficient Training of Graph-Regularized Multitask SVMs 633
Christian Widmer, Marius Kloft, Nico Görnitz, and Gunnar Rätsch

Geometry Preserving Multi-task Metric Learning 648
Peipei Yang, Kaizhu Huang, and Cheng-Lin Liu

Learning and Inference in Probabilistic Classifier Chains with Beam Search 665
Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan

Learning Multiple Tasks with Boosted Decision Trees 681
Jean Baptiste Faddoul, Boris Chidlovskii, Rémi Gilleron, and Fabien Torre

Multi-Task Boosting by Exploiting Task Relationships 697
Yu Zhang and Dit-Yan Yeung

Sparse Gaussian Processes for Multi-task Learning 711
Yuyang Wang and Roni Khardon

Natural Language Processing

Collective Information Extraction with Context-Specific Consistencies 728
Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe

Supervised Learning of Semantic Relatedness 744
Ran El-Yaniv and David Yanay

Unsupervised Bayesian Part of Speech Inference with Particle Gibbs 760
Gregory Dubbin and Phil Blunsom

WikiSent: Weakly Supervised Sentiment Analysis through Extractive Summarization with Wikipedia 774
Subhabrata Mukherjee and Pushpak Bhattacharyya

Online Learning and Data Streams

Adaptive Two-View Online Learning for Math Topic Classification 794
Tam T. Nguyen, Kuiyu Chang, and Siu Cheung Hui

BDUOL: Double Updating Online Learning on a Fixed Budget 810
Peilin Zhao and Steven C.H. Hoi

Handling Time Changing Data with Adaptive Very Fast Decision Rules 827
Petr Kosina and João Gama

Improved Counter Based Algorithms for Frequent Pairs Mining in Transactional Data Streams 843
Konstantin Kutzkov

Mirror Descent for Metric Learning: A Unified Approach 859
Gautam Kunapuli and Jude Shavlik

Author Index 875

Table of Contents – Part II

Privacy and Security

AUDIO: An Integrity *Audi*ting Framework of *O*utlier-Mining-as-a-Service Systems 1
Ruilin Liu, Hui (Wendy) Wang, Anna Monreale, Dino Pedreschi, Fosca Giannotti, and Wenge Guo

Differentially Private Projected Histograms: Construction and Use for Prediction 19
Staal A. Vinterbo

Fairness-Aware Classifier with Prejudice Remover Regularizer 35
Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma

Rankings and Recommendations

A Live Comparison of Methods for Personalized Article Recommendation at Forbes.com 51
Evan Kirshenbaum, George Forman, and Michael Dugan

Fast ALS-Based Tensor Factorization for Context-Aware Recommendation from Implicit Feedback 67
Balázs Hidasi and Domonkos Tikk

Probability Estimation for Multi-class Classification Based on Label Ranking 83
Weiwei Cheng and Eyke Hüllermeier

Reinforcement Learning and Planning

Adaptive Planning for Markov Decision Processes with Uncertain Transition Models via Incremental Feature Dependency Discovery 99
N. Kemal Ure, Alborz Geramifard, Girish Chowdhary, and Jonathan P. How

APRIL: Active Preference Learning-Based Reinforcement Learning 116
Riad Akrour, Marc Schoenauer, and Michèle Sebag

Autonomous Data-Driven Decision-Making in Smart Electricity Markets 132
Markus Peters, Wolfgang Ketter, Maytal Saar-Tsechansky, and John Collins

Bayesian Nonparametric Inverse Reinforcement Learning 148
Bernard Michini and Jonathan P. How

Bootstrapping Monte Carlo Tree Search with an Imperfect Heuristic 164
Truong-Huy Dinh Nguyen, Wee-Sun Lee, and Tze-Yun Leong

Fast Reinforcement Learning with Large Action Sets Using Error-Correcting Output Codes for MDP Factorization 180
Gabriel Dulac-Arnold, Ludovic Denoyer, Philippe Preux, and Patrick Gallinari

Learning Policies for Battery Usage Optimization in Electric Vehicles . . . 195
Stefano Ermon, Yexiang Xue, Carla Gomes, and Bart Selman

Policy Iteration Based on a Larned Transition Model 211
Vivek Ramavajjala and Charles Elkan

Structured Apprenticeship Learning 227
Abdeslam Boularias, Oliver Krömer, and Jan Peters

Rule Mining and Subgroup Discovery

A Bayesian Approach for Classification Rule Mining in Quantitative Databases 243
Dominique Gay and Marc Boullé

A Bayesian Scoring Technique for Mining Predictive and Non-Spurious Rules 260
Iyad Batal, Gregory Cooper, and Milos Hauskrecht

Generic Pattern Trees for Exhaustive Exceptional Model Mining 277
Florian Lemmerich, Martin Becker, and Martin Atzmueller

Semi-Supervised and Transductive Learning

Bidirectional Semi-supervised Learning with Graphs 293
Tomoharu Iwata and Kevin Duh

Coupled Bayesian Sets Algorithm for Semi-supervised Learning and Information Extraction 307
Saurabh Verma and Estevam R. Hruschka Jr.

Graph-Based Transduction with Confidence 323
Matan Orbach and Koby Crammer

Maximum Consistency Preferential Random Walks 339
Deguang Kong and Chris Ding

Semi-supervised Multi-label Classification: A Simultaneous Large-Margin, Subspace Learning Approach 355
Yuhong Guo and Dale Schuurmans

Sensor Data

MDL-Based Analysis of Time Series at Multiple Time-Scales 371
Ugo Vespier, Arno Knobbe, Siegfried Nijssen, and Joaquin Vanschoren

Separable Approximate Optimization of Support Vector Machines for Distributed Sensing 387
Sangkyun Lee, Marco Stolpe, and Katharina Morik

Unsupervised Inference of Auditory Attention from Biosensors 403
Melih Kandemir, Arto Klami, Akos Vetek, and Samuel Kaski

Sequence and String Mining

A Family of Feed-Forward Models for Protein Sequence Classification ... 419
Sam Blasiak, Huzefa Rangwala, and Kathryn B. Laskey

General Algorithms for Mining Closed Flexible Patterns under Various Equivalence Relations 435
Tomohiro I, Yuki Enokuma, Hideo Bannai, and Masayuki Takeda

Size Matters: Finding the Most Informative Set of Window Lengths 451
Jefrey Lijffijt, Panagiotis Papapetrou, and Kai Puolamäki

Social Network Mining

Discovering Links among Social Networks 467
Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino

Efficient Bi-objective Team Formation in Social Networks 483
Mehdi Kargar, Aijun An, and Morteza Zihayat

Feature-Enhanced Probabilistic Models for Diffusion Network Inference 499
Liaoruo Wang, Stefano Ermon, and John E. Hopcroft

Influence Spread in Large-Scale Social Networks – A Belief Propagation Approach 515
Huy Nguyen and Rong Zheng

Location Affiliation Networks: Bonding Social and Spatial Information 531
Konstantinos Pelechrinis and Prashant Krishnamurthy

On Approximation of Real-World Influence Spread 548
Yu Yang, Enhong Chen, Qi Liu, Biao Xiang, Tong Xu, and Shafqat Ali Shad

Opinion Formation by Voter Model with Temporal Decay Dynamics 565
Masahiro Kimura, Kazumi Saito, Kouzou Ohara, and Hiroshi Motoda

Viral Marketing for Product Cross-Sell through Social Networks 581
Ramasuri Narayanam and Amit A. Nanavati

Which Topic Will You Follow? 597
Deqing Yang, Yanghua Xiao, Bo Xu, Hanghang Tong, Wei Wang, and Sheng Huang

Spatial and Geographical Data Mining

Inferring Geographic Coincidence in Ephemeral Social Networks 613
Honglei Zhuang, Alvin Chin, Sen Wu, Wei Wang, Xia Wang, and Jie Tang

Pedestrian Quantity Estimation with Trajectory Patterns 629
Thomas Liebig, Zhao Xu, Michael May, and Stefan Wrobel

Socioscope: Spatio-temporal Signal Recovery from Social Media 644
Jun-Ming Xu, Aniruddha Bhargava, Robert Nowak, and Xiaojin Zhu

Statistical Methods and Evaluation

A Framework for Evaluating the Smoothness of Data-Mining Results ... 660
Gaurav Misra, Behzad Golshan, and Evimaria Terzi

Active Evaluation of Ranking Functions Based on Graded Relevance ... 676
Christoph Sawade, Steffen Bickel, Timo von Oertzen, Tobias Scheffer, and Niels Landwehr

Time Series and Temporal Data Mining

Community Trend Outlier Detection Using Soft Temporal Pattern Mining 692
Manish Gupta, Jing Gao, Yizhou Sun, and Jiawei Han

Data Structures for Detecting Rare Variations in Time Series 709
Caio Valentim, Eduardo S. Laber, and David Sotelo

Invariant Time-Series Classification 725
Josif Grabocka, Alexandros Nanopoulos, and Lars Schmidt-Thieme

Learning Bi-clustered Vector Autoregressive Models 741
Tzu-Kuo Huang and Jeff Schneider

Transfer Learning

Discriminative Factor Alignment across Heterogeneous Feature Space ... 757
Fangwei Hu, Tianqi Chen, Nathan N. Liu, Qiang Yang, and Yong Yu

Learning to Perceive Two-Dimensional Displays Using Probabilistic Grammars 773
Nan Li, William W. Cohen, and Kenneth R. Koedinger

Transfer Spectral Clustering 789
Wenhao Jiang and Fu-lai Chung

System Demonstrations Track

An Aspect-Lexicon Creation and Evaluation Tool for Sentiment Analysis Researchers 804
Mus'ab Husaini, Ahmet Koçyiğit, Dilek Tapucu, Berrin Yanikoglu, and Yücel Saygın

Association Rule Mining Following the Web Search Paradigm 808
Radek Škrabal, Milan Šimůnek, Stanislav Vojíř, Andrej Hazucha, Tomáš Marek, David Chudán, and Tomáš Kliegr

ASV Monitor: Creating Comparability of Machine Learning Methods for Content Analysis 812
Andreas Niekler, Patrick Jähnichen, and Gerhard Heyer

ClowdFlows: A Cloud Based Scientific Workflow Platform 816
Janez Kranjc, Vid Podpečan, and Nada Lavrač

Extracting Trajectories through an Efficient and Unifying Spatio-temporal Pattern Mining System 820
Phan Nhat Hai, Dino Ienco, Pascal Poncelet, and Maguelonne Teisseire

Knowledge Discovery through Symbolic Regression with HeuristicLab 824
Gabriel Kronberger, Stefan Wagner, Michael Kommenda, Andreas Beham, Andreas Scheibenpflug, and Michael Affenzeller

OutRules: A Framework for Outlier Descriptions in Multiple Context Spaces 828
Emmanuel Müller, Fabian Keller, Sebastian Blanc, and Klemens Böhm

Scientific Workflow Management with ADAMS 833
Peter Reutemann and Joaquin Vanschoren

TopicExplorer: Exploring Document Collections with Topic Models 838
Alexander Hinneburg, Rico Preiss, and René Schröder

VIKAMINE – Open-Source: Subgroup Discovery, Pattern Mining, and Analytics 842
Martin Atzmueller and Florian Lemmerich

Nectar Track

Learning Submodular Functions 846
Maria-Florina Balcan and Nicholas J.A. Harvey

Matrix Factorization as Search 850
Kristian Kersting, Christian Bauckhage, Christian Thurau, and Mirwaes Wahabzada

Metal Binding in Proteins: Machine Learning Complements X-Ray Absorption Spectroscopy 854
Marco Lippi, Andrea Passerini, Marco Punta, and Paolo Frasconi

Modelling Input Varying Correlations between Multiple Responses 858
Andrew Gordon Wilson and Zoubin Ghahramani

Author Index 863

Machine Learning for Robotics

Pieter Abbeel

University of California, Berkeley, USA
pabbeel@cs.berkeley.edu
http://www.cs.berkeley.edu/~pabbeel/

Abstract. Robots are typically far less capable in autonomous mode than in teleoperated mode. The few exceptions tend to stem from long days (and more often weeks, or even years) of expert engineering for a specific robot and its operating environment. Current control methodology is quite slow and labor intensive. I believe advances in machine learning have the potential to revolutionize robotics. In this talk, I will present new machine learning techniques we have developed that are tailored to robotics. I will describe in depth "Apprenticeship learning", a new approach to high-performance robot control based on learning for control from ensembles of expert human demonstrations. Our initial work in apprenticeship learning has enabled the most advanced helicopter aerobatics to-date, including maneuvers such as chaos, tic-tocs, and auto-rotation landings which only exceptional expert human pilots can fly. Our most recent work in apprenticeship learning is providing traction on learning to perform challenging robotic manipulation tasks, such as knot-tying. I will also briefly highlight three other machine learning for robotics developments: Inverse reinforcement learning and its application to quadruped locomotion, Safe exploration in reinforcement learning which enables robots to learn on their own, and Learning for perception with application to robotic laundry.

Bio

Pieter Abbeel received a BS/MS in Electrical Engineering from KU Leuven (Belgium) and received his Ph.D. degree in Computer Science from Stanford University in 2008. He joined the faculty at UC Berkeley in Fall 2008, with an appointment in the Department of Electrical Engineering and Computer Sciences. He has won various awards, including best paper awards at ICML and ICRA, the Sloan Fellowship, the Air Force Office of Scientific Research Young Investigator Program (AFOSR-YIP) award, the Okawa Foundation award, the 2011s TR35, and the IEEE Robotics and Automation Society (RAS) Early Career Award. He has developed apprenticeship learning algorithms which have enabled advanced helicopter aerobatics, including maneuvers such as tic-tocs, chaos and auto-rotation, which only exceptional human pilots can perform. His group has also enabled the first end-to-end completion of reliably picking up a crumpled laundry article and folding it. His work has been featured in many popular press outlets, including BBC, New York Times, MIT Technology Review, Discovery Channel, SmartPlanet and Wired. His current research focuses on robotics and machine learning with a particular focus on challenges in personal robotics, surgical robotics and connectomics.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, p. 1, 2012.

Declarative Modeling for Machine Learning and Data Mining

Luc De Raedt

University of Leuven, Belgium
luc.deraedt@cs.kuleuven.be
http://people.cs.kuleuven.be/~luc.deraedt/

Abstract. Despite the popularity of machine learning and data mining today, it remains challenging to develop applications and software that incorporates machine learning or data mining techniques. This is because machine learning and data mining have focussed on developing high-performance algorithms for solving particular tasks rather than on developing general principles and techniques. I propose to alleviate these problems by applying the constraint programming methodology to machine learning and data mining and to specify machine learning and data mining problems as constraint satisfaction and optimization problems. What is essential is that the user be provided with a way to declaratively specify what the machine learning or data mining problem is rather than having to outline how that solution needs to be computed. This corresponds to a model + solver-based approach to machine learning and data mining, in which the user specifies the problem in a high level modeling language and the system automatically transforms such models into a format that can be used by a solver to efficiently generate a solution. This should be much easier for the user than having to implement or adapt an algorithm that computes a particular solution to a specific problem. Throughout the talk, I shall use illustrations from our work on constraint programming for itemset mining and probabilistic programming.

Bio

Luc De Raedt is a full professor (of research) at the University of Leuven (KU Leuven) in the Department of Computer Science and a former chair of Machine Learning at the Albert-Ludwigs-University in Freiburg. Luc De Raedt has been working in the areas of artificial intelligence and computer science, especially on computational logic, machine learning and data mining, probabilistic reasoning and constraint programming and their applications in bio- and chemoinformatics, vision and robotics, natural language processing, and engineering. His work has typically crossed boundaries between different research areas, often working towards an integration of their principles. He is well-known for his early work on inductive logic programming (combining logic with learning). Since 2000, he has been working towards a further integration of logical and relational learning with probabilistic reasoning (statistical relational learning and probabilistic programming) and on inductive querying in databases. During the last three

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 2–3, 2012.

years he has been fascinated by the possibility of combining constraint programming principles with data mining and machine learning. He is currently coordinating a European IST FET project in this area (ICON Inductive Constraint Programming) and is the program chair of the 20th European Conference on Artificial Intelligence (Montpellier, 2012). He was a program co-chair of ICML 2005 and ECML/PKDD 2001.

Machine Learning Methods for Music Discovery and Recommendation

Douglas Eck

Google Research
douglas.eck@gmail.com
research.google.com/

Abstract. In this talk I will relate current work at Google in music recommendation to the challenge of automatic music annotation ("autotagging"). I will spend most of the talk looking at (a) signal processing and sparse coding strategies for pulling relevant structure from audio, and (b) training multi-class ranking models in order to build good music similarity spaces. Although I will describe some technical aspects of autotagging and ranking via embedding, the main goal of the talk is to foster a better understanding of the real-world challenges we face in helping users find music they'll love. To this end I will play a number of audio demos illustrating what we can (and cannot) hope to achieve by working with audio.

Bio

Douglas Eck is a research scientist at Google in Mountain View, California. His current focus is on machine learning models and user interfaces for music discovery and recommendation. This involves not only algorithm development but also user studies and data analyses to better understand what listeners want from a music service. Before coming to Google, Douglas was an associate professor in Computer Science at University of Montreal where he worked in related areas such as meter and beat induction, automatic tagging of music tracks and expressive timing and dynamics in music performance.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, p. 4, 2012.

Solving Problems with Visual Analytics: Challenges and Applications

Daniel Keim

University of Konstanz
Daniel.Keim@uni-konstanz.de
www.informatik.uni-konstanz.de/arbeitsgruppen/infovis/mitglieder/prof-dr-daniel-keim/

Abstract. Never before in history data is generated and collected at such high volumes as it is today. As the volumes of data available to business people, scientists, and the public increase, their effective use becomes more challenging. Keeping up to date with the flood of data, using standard tools for data analysis and exploration, is fraught with difficulty. The field of visual analytics seeks to provide people with better and more effective ways to explore and understand large datasets, while also enabling them to act upon their findings immediately. Visual analytics integrates the analytic capabilities of the computer and the perceptual and intellectual abilities of the human analyst, allowing novel discoveries and empowering individuals to take control of the analytical process. Visual analytics enables unexpected insights, which may lead to beneficial and profitable innovation. The talk presents the challenges of visual analytics and exemplifies them with several application examples, which illustrate the exiting potential of current visual analysis techniques but also their limitations.

Bio

Daniel A. Keim is full professor and head of the Information Visualization and Data Analysis Research Group at the University of Konstanz, Germany. He has been actively involved in information visualization and data analysis research for about 20 years and developed a number of novel visual analysis techniques for very large data sets with applications to a wide range of application areas including financial analysis, network analysis, geo-spatial analysis, as well as text and multimedia analysis. His research resulted in two recent books “Solving problems with Visual Analytics” and “Interactive Data Visualization” which he both co-authored. Dr. Keim has been program co-chair of the IEEE InfoVis and IEEE VAST symposia as well as the SIGKDD conference, and he is or was member of the IEEE InfoVis, IEEE VAST, and EuroVis steering committees. He is an associate editor of Palgrave’s Information Visualization Journal (since 2001) and has been an associate editor of the IEEE Transactions on Visualization and Computer Graphics (1999–2004), the IEEE Transactions on Knowledge and Data Engineering (2002–2007), and the Knowledge and Information System Journal (2006–2011). He is coordinator of the German Strategic Research Initiative (SPP) on

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 5–6, 2012.

Scalable Visual Analytics and he was the scientific coordinator of the EU Coordination Action on Visual Analytics called VisMaster. Dr. Keim got his Ph.D. and habilitation degrees in computer science from the University of Munich. Before joining the University of Konstanz, Dr. Keim was associate professor at the University of Halle, Germany and Technology Consultant at AT&T Shannon Research Labs, NJ, USA.

Analyzing Text and Social Network Data with Probabilistic Models

Padhraic Smyth

University of California, Irvine, USA
smyth@ics.uci.edu
http://www.ics.uci.edu/~smyth/

Abstract. Exploring and understanding large text and social network data sets is of increasing interest across multiple fields, in computer science, social science, history, medicine, and more. This talk will present an overview of recent work using probabilistic latent variable models to analyze such data. Latent variable models have a long tradition in data analysis and typically hypothesize the existence of simple unobserved phenomena to explain relatively complex observed data. In the past decade there has been substantial work on extending the scope of these approaches from relatively small simple data sets to much more complex text and network data. We will discuss the basic concepts behind these developments, reviewing key ideas, recent advances, and open issues. In addition we will highlight common ideas that lie beneath the surface of different approaches including links (for example) to work in matrix factorization. The concluding part of the talk will focus more specifically on recent work with temporal social networks, specifically data in the form of time-stamped events between nodes (such as emails exchanged among individuals over time).

Bio

Padhraic Smyth is a Professor at the University of California, Irvine, in the Department of Computer Science with a joint appointment in Statistics, and is also Director of the Center for Machine Learning and Intelligent Systems at UC Irvine. His research interests include machine learning, data mining, pattern recognition, and applied statistics and he has published over 150 papers on these topics. He was a recipient of best paper awards at the 2002 and 1997 ACM SIGKDD Conferences, received the ACM SIGKDD Innovation Award in 2009, and was named a AAAI Fellow in 2010. He is co-author of Modeling the Internet and the Web: Probabilistic Methods and Algorithms (with Pierre Baldi and Paolo Frasconi in 2003), and co-author of Principles of Data Mining, MIT Press (with David Hand and Heikki Mannila in 2001). Padhraic has served in editorial and advisory positions for journals such as the Journal of Machine Learning Research, the Journal of the American Statistical Association, and the IEEE Transactions on Knowledge and Data Engineering. While at UC Irvine he has received research funding from agencies such as NSF, NIH, IARPA, NASA, and DOE, and from companies such as Google, IBM, Yahoo!, Experian, and Microsoft. In addition to his academic research he is also active in industry consulting, working with companies such as eBay, Yahoo!, Microsoft, Oracle, Nokia, and AT&T, as well as serving as scientific advisor to

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 7–8, 2012.

local startups in Orange County. He also served as an academic advisor to Netflix for the Netflix prize competition from 2006 to 2009. Padhraic received a first class honors degree in Electronic Engineering from National University of Ireland (Galway) in 1984, and the MSEE and PhD degrees (in 1985 and 1988 respectively) in Electrical Engineering from the California Institute of Technology. From 1988 to 1996 he was a Technical Group Leader at the Jet Propulsion Laboratory, Pasadena, and has been on the faculty at UC Irvine since 1996.

Discovering Descriptive Tile Trees
By Mining Optimal Geometric Subtiles

Nikolaj Tatti and Jilles Vreeken

Advanced Database Research and Modeling
Universiteit Antwerpen
{nikolaj.tatti,jilles.vreeken}@ua.ac.be

Abstract. When analysing binary data, the ease at which one can interpret results is very important. Many existing methods, however, discover either models that are difficult to read, or return so many results interpretation becomes impossible. Here, we study a fully automated approach for mining easily interpretable models for binary data. We model data hierarchically with noisy tiles—rectangles with significantly different density than their parent tile. To identify good trees, we employ the Minimum Description Length principle.

We propose Stijl, a greedy any-time algorithm for mining good tile trees from binary data. Iteratively, it finds the locally *optimal* addition to the current tree, allowing overlap with tiles of the same parent. A major result of this paper is that we find the optimal tile in only $\Theta(NM \min(N, M))$ time. Stijl can either be employed as a top-k miner, or by MDL we can identify the tree that describes the data best.

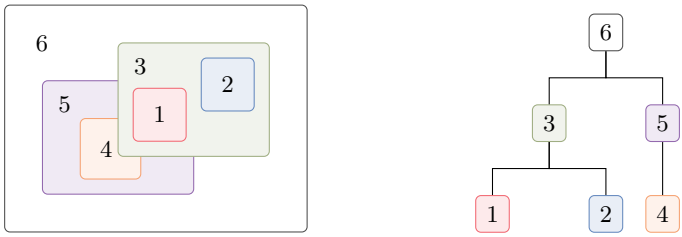
Experiments show we find succinct models that accurately summarise the data, and, by their hierarchical property are easily interpretable.

1 Introduction

When exploratively analysing a large binary dataset, being able to easily interpret the results is of utmost importance. Many data analysis methods, however, have trouble meeting this requirement. With frequent pattern mining, for example, we typically find overly many and highly redundant results, hindering interpretation [10]. Pattern set mining [2,5,21] tackles these problems, and instead provides small and high-quality sets of patterns. However, as these methods generally exploit complex statistical dependencies between patterns, the resulting models are often difficult to fully comprehend.

When analysing 0–1 data, the encompassing question is 'how are the 1s distributed?'. In this paper, we focus on the underlying questions of 'where are the ones?' and 'where are the zeroes?'. To answer these questions in an easily interpretable manner, we propose to model the data hierarchically, by identifying trees of tiles, i.e. sub-matrices that are surprisingly dense or sparse compared to their parent tile. As an example, consider Figure 1, in which we show a toy example of a hierarchical tiling, and the corresponding tile tree. As the figure shows, tiles model parts of the data, and subtiles provide refinements over their

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 9–24, 2012.



(a) Hierarchical tiling (b) Tile tree

Fig. 1. Toy example of a tiled database, and the corresponding tile tree structure

parents. Next, as an example on real data, consider Figure 2, in which we show the tiling our algorithm discovered on paleontological data. Very easily read, using only 14 tiles, the tiling shows which regions of the data are relatively dense (dark), as well as where relatively few 1s are found (light).

Clearly, we aim to mine descriptions that are succinct, non-redundant, and neither overly complex nor simplistic. We therefore formalise the problem in terms of the Minimum Description Length (MDL) principle [9], by which we can automatically identify the model that best describes the data, without having to set any parameters. For mining good models, we introduce Stijl, a heuristic any-time algorithm that iteratively greedily finds the optimal subtile and adds it to the current tiling. A major result of this paper is that we show that we can find such optimal subtiles in only $\Theta(NM \min(N, M))$, as opposed to $\Theta(N^2M^2)$ when done naively [8].

We are not the first to study the problem of hierarchical tiling. The problem was first introduced by Gionis et al. [8], whom proposed a randomised approach as an alternative to the naive approach. Our FindTile procedure, on the other hand, is deterministic and identifies *optimal* subtiles. Moreover, our MDL formalisation requires no scaling parameters, making the method parameter-free.

These differences aside, both methods assume an order on the rows and columns of the data; as for such data, a subtile can be straightforwardly defined by a 'from' and 'to' selection query. As such, we exploit that the data is ordered, as this allows us to generate more easily understandable and easily visually representable models for the data. Although many datasets naturally exhibit such

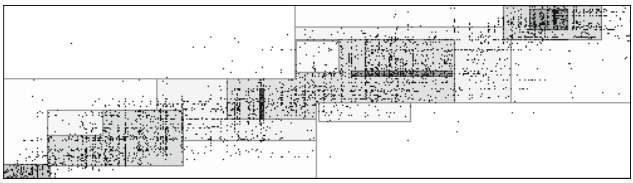


Fig. 2. Tiling of the *Paleo* dataset. See Fig. 4(b) for a cleaned version without 1s

order, e.g. spatially and/or temporally, not all data does. For unordered data, e.g. through spectral ordering, good orders can be discovered [6,8,19].

Experimentation on our method shows we discover easily interpretable models that describe the data very well. Stijl mines trees that summarise the data succinctly, with non-redundant tile trees that consist of relatively few tiles.

The paper is organised as follows. Section 2 discusses preliminaries. Section 3 gives the Stijl algorithm for mining tile trees, and Section 4 details mining optimal subtiles. We discuss related work in Section 5, and experiment in Section 6. We round up with discussion and conclusions. Due to lack of space, we give the proofs for Propositions 1–2 in the Appendix[1]

2 Encoding Data with Tile Trees

We begin by giving the basic definitions we use throughout the paper, after which we discuss how we can measure the quality of a hierarchical tile set.

Notation. A *binary dataset* D is a binary matrix of size N-by-M consisting of N rows, binary vectors of size M. We denote $(i,j)^{\text{th}}$ entry of D by $D(i,j)$. We assume that both rows and columns have an order and, for simplicity, we assume that the indexing corresponds to the orders.

A geometric *tile* $X = (a,b) \times (c,d)$, where $1 \leq a \leq b \leq N$ and $1 \leq c \leq d \leq M$, identifies a consecutive submatrix of D. In contrast, for combinatorial tiles, the rows and columns are not required to be consecutive. In this paper, we focus on geometric tiles. We say that $X_1 = (a_1,b_1) \times (c_1,d_1)$ is a subtile of $X_2 = (a_2,b_2) \times (c_2,d_2)$ if X_1 is completely covered by X_2, that is, $a_2 \leq a_1$, $b_1 \leq b_2$, $c_2 \leq c_1$, and $d_1 \leq d_2$. We will write $(i,j) \in X$ if $a \leq i \leq b$ and $c \leq j \leq d$.

A *tile tree* $\mathcal{T}$ is a tree of tiles such that each child of a tile $X \in \mathcal{T}$ is a subtile of X. We will denote the children of X by $\mathit{children}(X)$. In our setting, the order of the children matters, so we assume that $\mathit{children}(X)$ is a *list* of tiles and not a set. We also assume that the root tile always covers the whole data. Given a tile tree $\mathcal{T}$, a tile $X \in \mathcal{T}$ and a subtile Y of X, we will write $\mathcal{T} + X \to Y$ to denote a tile tree obtained by adding Y as a last child of X.

Our next step is to define which data entries are covered by which tile. Since we allow child tiles to overlap, the definition is involved—although intuition is simple: the first most-specific tile that can encode a cell, encodes its value, and all other tiles ignore it. More formally, given a tile tree $\mathcal{T}$, consider a post-order, that is, an order where the child tiles appear before their parents and such that if $\mathit{children}(X) = (Y_1, \ldots, Y_L)$, then Y_i is appears before Y_{i+1}. Let $X \in \mathcal{T}$. We define $\mathit{tid}(X;\mathcal{T})$ to be the position of X in the post-order. When $\mathcal{T}$ is clear from the context we will simply write $\mathit{tid}(X)$. An example of the post-order is given in Figure 1(b). Using this order we can define which entries belong to which tile. We define

$$\mathit{area}(X;\mathcal{T}) = \{(i,j) \in X \mid \text{there is no } Y \text{ with } (i,j) \in Y, \mathit{tid}(Y) < \mathit{tid}(X)\},$$

[1] http://adrem.ua.ac.be/stijl/

that is, entries are assigned to the cells first-come first-serve, see Figure 1(a) as an example. Among these entries, we define the number of 1s and 0s as

$$p(X;\mathcal{T},D) = |\{(i,j) \in \mathit{area}(X;\mathcal{T}) \mid D(i,j) = 1\}| \quad \text{and}$$
$$n(X;\mathcal{T},D) = |\{(i,j) \in \mathit{area}(X;\mathcal{T}) \mid D(i,j) = 0\}| \quad .$$

Let us denote by $|\mathcal{T}|$ the number of tiles in a tree $\mathcal{T}$, i.e. $|\mathcal{T}| = |\{X \in \mathcal{T}\}|$, and denote by $\mathcal{T}_0$ the most simple tile tree consisting of only a root tile.

MDL for Tile Trees. Our main goal is to find tile trees that summarise the data well; they should be succinct yet highly informative on where the 1s on the data are. We can formalise this intuition through the Minimum Description Length (MDL) principle [9], a practical version of Kolmogorov Complexity [13]. Both embrace the slogan *Induction by Compression*. The MDL principle can be roughly described as follows: Given a dataset D and a set of models $\mathcal{X}$ for D, the best model $X \in \mathcal{X}$ is the one that minimises $L(X) + L(D \mid X)$ in which $L(X)$ is the length, in bits, of the description of the model X, and $L(D \mid X)$ is the length, in bits, of the data as described using X.

This is called two-part MDL, or *crude* MDL. This stands opposed to *refined* MDL, where model and data are encoded together [9]. We use two-part MDL because we are specifically interested in the model: the tile tree $\mathcal{T}^*$ that yields the minimal description length. Further, although refined MDL has stronger theoretical foundations, it cannot be computed except for some special cases [9]. Before we can use MDL to identify good models, we will have to define how to encode a database given a tile tree, as well as how to encode a tile tree.

We encode the values of $\mathit{area}(X;\mathcal{T})$ using prefix codes. The length of an optimal prefix code is given by Shannon entropy, i.e. $-\log P(\cdot)$, where $P(\cdot)$ is the probability of a value [4]. We have the optimal encoded length for all entries $\mathit{area}(X;\mathcal{T})$ of a tile X in a tile tree $\mathcal{T}$ as

$$L(D \mid X,\mathcal{T}) = L(p(X;\mathcal{T},D), n(X;\mathcal{T},D)),$$

where $L(p,n) = -p\log\frac{p}{p+n} - n\log\frac{n}{p+n}$ is the scaled entropy.

In order to compare fairly between models, MDL requires the encoding to be lossless. Hence, besides the data, we also have to encode the tile tree itself.

We encode tile trees node per node, in reverse order, and add extra bits between the tiles to indicate the tree structure. We use a bit of value 1 to indicate that the next tile is a child of the current tile, and 0 to indicate that we have processed all child tiles of the current tile. For example, the tree given in Figure 1(b) is encoded, with <tile i> indicating an encoded tile, as

<tile 6>1<tile 5>1<tile 4>001<tile 3>1<tile 2>01<tile 1>000 .

To encode an individual tile, we proceed as follows. Let X be a non-root tile and let $Z = (a,b) \times (c,d)$ be the direct parent tile of X. As we know that X is a subtile of Z, we know the end points for defining the area of X fall within those of Z. As such, to encode the 4 end points of X we need only $4\log(b-a+1) + 4\log(d-c+1)$ bits.

We also know that number of 1s in X are bounded by the area of Z, $(b-a+1)(d-c+1)$, and hence we can encode the number of 1s in X in $\log(b-a+1)+\log(d-c+1)$ bits. Note that although we can encode the number of 1s more efficiently by using the geometry of X instead of Z, this would introduce a bias to small tiles.

Next, to calculate the encoded size of a tile, we need to take the two bits for encoding the tree structure of X into account. As describe above, one bit is used to indicate that X has no more children and the other to indicate that X is a child of Z. Putting this together, the encoded length of a non-root tile X is

$$L(X \mid \mathcal{T}) = 1 + 1 + 5\log(b-a+1) + 5\log(d-c+1) \quad .$$

Let us now assume that X is the root tile. Since we require that a root tile covers the whole data, we need to encode the dimensions of the data set, the number of 1s in X, and following 1 bit to indicate that all tiles have been processed. Unlike for the other tiles in the tree, we have no upper bound for the dimensions of X, and therefore would have to use a so-called Universal Code [9] to encode the dimensions—after which we could subsequently encode the number of 1s in X in $\log N + \log M$ bits. However, as the lengths of these codes are constant over all models for D, and we can safely ignore them when selecting between models. For simplicity, for a root tile X we therefore define $L(X \mid \mathcal{T}) = 0$.

As such, we have for the total encoded size of a database D and a tile tree $\mathcal{T}$

$$L(D, \mathcal{T}) = \sum_{X \in \mathcal{T}} L(X \mid \mathcal{T}) + L(D \mid X, \mathcal{T}) \quad ,$$

by which we now have a formal MDL score for tile trees.

3 Mining Good Tile Trees

Now that we have defined how to encode data with a tile tree, our next step is to find the best tile tree, i.e. the tile tree minimising the total encoded length. That is, we want to solve the following problem.

Problem 1 (Minimal Tile Tree). Given a binary dataset D, find a tile tree $\mathcal{T}$ such that the total encoded size, $L(D, \mathcal{T})$, is minimised.

As simply as it is stated, this minimisation problem is rather difficult to solve. Besides that the search space of all possible tile trees is rather vast, the total encoded size $L(D, \mathcal{T})$ does not exhibit trivial structure that we can exploit for fast search, e.g. (weak) monotonicity. Hence, we resort to heuristics.

For finding an approximate solution to the Minimal Tile Tree problem, we propose the Stijl algorithm.[2] We give the pseudo-code as Algorithm 1. We iteratively find that subtile Y of a tile $X \in \mathcal{T}$ by which the total encoded size is minimised. We therefore refer to Y as the optimal subtile of X. After identifying

[2] Named after the art movement De Stijl, to which art our models show resemblance.

Algorithm 1: STIJL$(D, \mathcal{T}, X)$

```
input  : dataset D, current tile tree T, parent tile X
output : updated tile tree T
Y ← subtile of X minimising L(D, T + X → Y);
while L(D, T + X → Y) < L(D, T) do
    T ← STIJL(D, T + X → Y, Y);
    Y ← subtile of X minimising L(D, T + X → Y);
return T;
```

the optimal subtile, STIJL adds Y into the tile tree, and continues inductively until no improvement can be made.

Alternative to this approach, we can also approximate the optimal k-tile tree. To do so, we adapt the algorithm to find the subtile Y over all parent tiles $X \in \mathcal{T}$ that minimises the score—as opposed to our standard depth-first strategy. Note that by the observation above, for the k at which the score is minimised, both strategies find the same tree.

By employing a greedy heuristic, we have reduced the problem of finding the optimal tile tree into a problem of finding the optimal subtile.

Problem 2 (Minimal Subtile). Given a binary dataset D, a tile tree $\mathcal{T}$, and a tile $X \in \mathcal{T}$, find a tile Y such that Y is a subtile of X, and $\mathcal{T} + X \to Y$ is minimised.

The main part of this paper details how to find an optimal subtile, a procedure we subsequently use in STIJL.

4 Finding the Optimal Tile

In this section we focus on finding the optimal subtile. Naively, we solve this by simply testing every possible subtile, requiring $\Theta(N^2M^2)$ tests, where N and M are the number of rows and columns in the parent tile, respectively [8].

In this section, we present an algorithm that can find the optimal subtile in $\Theta(N^2M)$. In order to do that, we will break the problem into two subproblems. The first problem is that for two *fixed* integers $c \leq d$, we need to find two integers $a \leq b$ such that the tile $(a, b) \times (c, d)$ is optimal. Once we have solved this, we can proceed to find the optimal tile by finding the optimal (c, d).

We begin by giving an easier formulation of the function we want to optimise. In order to do so, note that adding a subtile Y to X changes only $area(T; \mathcal{X})$, and does not influence (the encoded length of) other tiles in the tree. Hence, we expect to be able to express the difference in total encoded length between $\mathcal{T} + X \to Y$ and $\mathcal{T}$ in simple terms. In fact, we have the following theorem.

Proposition 1. *Let $\mathcal{T}$ be a tile tree. Let $X \in \mathcal{T}$ be a tile and let Y be a subtile of X. Define $\mathcal{T}' = \mathcal{T} + X \to Y$ and have $u = p(Y; \mathcal{T}')$, $v = n(Y; \mathcal{T}')$, $o = p(X; \mathcal{T})$, and $z = n(X; \mathcal{T})$. Then*

$$L(D, \mathcal{T}') - L(D, \mathcal{T}) = L(u, v) + L(o - u, z - v) - L(o, z) + L(Y \mid \mathcal{T}') \ .$$

In order to find the optimal subtile it is enough to create an algorithm for finding an optimal subtile more dense than its parent tile. To see this, note that we can find the optimal tile by first finding the optimal *dense* tile, and then find the optimal *sparse* tile by applying the same algorithm on the 0–1 inverse of the data. Once we have both the optimal optimal dense and optimal sparse tiles, we can choose the overall optimal subtile by MDL.

Let $X = (s, e) \times (x, y)$ be a tile, and $\mathcal{T}$ a tile tree with $X \in \mathcal{T}$. Assume that we are given indices c and d. Our goal in this section is to find those indices a and b such that $Y = (a, b) \times (c, d)$ is an optimal subtile of X.

Define two vectors, p for positives and n for negatives, each of length $e - s + 1$, that contain the number of 1s and 0s respectively, within the ith row of X, $p_i = |\{(i + s - 1, w) \in area(X) \mid c \leq w \leq d, D(i + s - 1, w) = 1\}|$, and $n_i = |\{(i + s - 1, w) \in area(X) \mid c \leq w \leq d, D(i + s - 1, w) = 0\}|$.

This allows us to define $cnt(a, b; p) = \sum_{i=a}^{b} p_i$ (and similarly for n). Let $u = cnt(a, b; p)$ and $v = cnt(a, b; n)$. It follows that $p(Y; \mathcal{T} + X \to Y) = u$ and $n(Y; \mathcal{T} + X \to Y) = v$, where $Y = (a, b) \times (c, d)$; those are the entries of X now to be encoded by Y. Let us define $cost(a, b; p, n, o, z) = L(u, v) + L(o - u, z - v)$. We will write $cost(a, b)$, when p, n, o, z are known from the context. Proposition 1 states that minimising $L(D, \mathcal{T}')$ is equivalent to minimising $cost(a, b)$.

Further, let us define $fr(a, b; p, n) = cnt(a, b; p) / (cnt(a, b; p) + cnt(a, b; n))$ to be the *frequency* of 1s within Y. Proposition 1 then allows us to formulate the optimisation problem as follows.

Problem 3 (Minimal Border Points). Let p and n be two integer vectors of the same length, m. Let o and z be two integers such that $cnt(1, m; p) \leq o$ and $cnt(1, m; n) \leq z$. Find $1 \leq a \leq b \leq n$ such that $fr(a, b; p, n) > o/(o+z)$ and that $cost(a, b)$ is minimised.

The rest of the section is devoted to solving this optimisation problem. Naively we could test every pair (a, b), which however requires quadratic time. Our approach is to ignore a large portion of suboptimal pairs, such that our search becomes linear.

To this end, let p and n be two vectors, and let $1 \leq b \leq |p|$ be an integer. We say that $a \leq b$ is a *head border* of b if there are no integers i and j such that $1 \leq i < a \leq j \leq b$ and $fr(i, a - 1) \geq fr(a, b)$. Similarly, we say that $b \geq a$ is a *tail border* of a if there are no indices $a \leq i \leq b < j \leq |p|$ such that $fr(i, b) \leq fr(b + 1, j)$. We denote the list of all head borders by $bh(b, p, n)$ and the list of all tail borders by $bt(a, p, n)$.

Given a head border a of b, we say that a is a *head candidate* if there are no indices $1 \leq i < a \leq j \leq b$ such that $fr(i, a - 1) \geq fr(j, b)$. Similarly, we say that $b \in bt(a)$ is a *tail candidate* of a if there are no indices $a \leq i \leq b < j \leq |p|$ such that $fr(a, i) \leq fr(b + 1, j)$. We denote the list of all head candidates by $ch(b, p, n)$ and the list of all tail candidates by $ct(a, p, n)$.

To avoid clutter, we do not write p and n wherever clear from context.

As an example, consider Figure 3(a). Since $fr(a_2, a_3 - 1) > fr(a_3, a_4 - 1)$, it follows that $a_3 \notin bh(a_4 - 1)$. Note that $a_4 \in bh(a_6 - 1)$ but $a_4 \notin ch(a_6 - 1)$ since $fr(a_2, a_4 - 1) > fr(a_5, a_6 - 1)$.

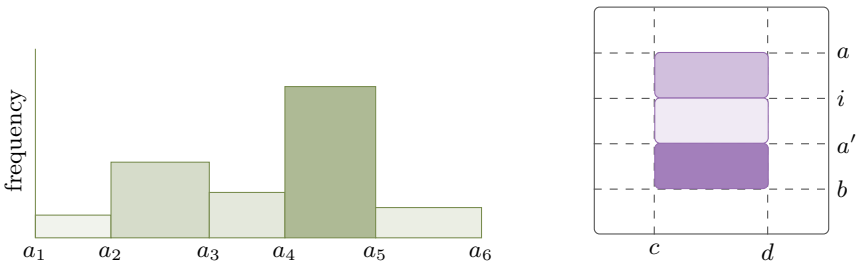


Fig. 3. Example of how FindTile considers head candidates. (a) a_3 is no head border for $a_4 - 1$, as $fr(a_2, a_3 - 1) > fr(a_3, a_4 - 1)$. Although a head border for $a_6 - 1$, a_4 is not a head candidate for $a_6 - 1$, as $fr(a_3, a_4 - 1) > fr(a_5, a_6 - 1)$. (b) Proposition 2 states that we can ignore i as head candidate for a tile to j, as by $fr(u', i - 1) > fr(i, u - 1)$ we know $fr(u', j - 1) > fr(i, j - 1)$.

We are now ready to state the main result of this section: in order to find the optimal tile we need to only study head and tail candidates.

Proposition 2. *Let p and n be two vectors and let o and z be two integers. Let $i \leq j$ be two indices such that $fr(i, j) > o/(o + z)$. Then there are $a \leq b$ such that $cost(a, b) \leq cost(i, j)$, $a \in ch(b)$ and $b \in ct(a)$.*

Proposition 2 is illustrated in Figure 3(b). Since $fr(a, i - 1) \geq fr(i, a')$, we know that $i \notin ch(b)$. Proposition 2 implies that we can safely ignore (i, b) and consider instead (a', b) or (a, b).

Proposition 2 states that we need to only study candidates, a subset of borders. Fortunately, there exists an efficient algorithm to construct a border list $bh(b + 1)$ given the existing list $bh(b)$ [3]. The approach relies on several lemmata.

Let $(a_1, \ldots, a_K) = bh(b)$. We claim that $bh(b + 1) \subseteq (a_1, \ldots, a_K, b + 1)$.

Lemma 1. *If $a \leq b$ and $a \notin bh(b)$, then $a \notin bh(b + 1)$.*

Proof. By definition there are i and j such that $1 \leq i < a \leq j \leq b$ such that $fr(i, a - 1) \geq fr(a, j)$. These indices are valid for $b + 1$, hence $a \notin bh(b + 1)$. □

Hence, in order to construct $bh(b + 1)$, we only need to delete entries from $(a_1, \ldots, a_K, b + 1)$. Let us define a *head frequency* $hfr(b) = \max_{i \leq b} fr(i, b)$ and a *tail frequency* $tfr(a) = \max_{a \geq i} fr(a, i)$. The following two lemmata say that the last entry in $bh(b + 1)$ has to be the smallest index j such that $fr(j, b) = hfr(b)$, and that the borders of $b + 1$ smaller than j are all included in $bh(b)$.

Lemma 2. *Let j be the smallest index s.t. $fr(j, b) = hfr(b)$. Then $j = \max bh(b)$. Let j be the largest index s.t. $fr(a, j) = tfr(a)$. Then $j = \min bt(a)$.*

Proof. First note that $j \in bh(b)$. Let i be an index $j < i \leq b$. We have $fr(i, b) \leq fr(j, b)$ which implies that $fr(j, i - 1) \geq fr(i, b)$. This implies that $i \notin bh(b)$. The case for $bt(a)$ is similar. □

Algorithm 2: SCAN(p, n, o, z)

```
   input  : integer vectors p and n, number of 1s (0s) in the parent tile, o (z)
   output : an interval t solving Problem 3
 1 best ← ∞; t ← (0, 0); B ← C ← ∅;
 2 foreach b = 1, ..., |p| do
 3     push b to the front of B;
 4     push b to the front of C;
 5     while |B| > 1 and fr(B_1, b; p, n) ≤ fr(B_2, B_1 − 1; p, n) do
 6         if B_1 = C_1 then remove C_1;
 7         remove B_1;
 8     while |C| > 1 and fr(C_2, C_1 − 1; p, n) ≥ tfr(b + 1) do
 9         c ← cost(C_1, b);
10         if c < best then t ← (C_1, b); best ← c;
11         remove C_1;
12     c ← cost(C_1, b);
13     if c < best then t ← (C_1, b); best ← c;
14 return t;
```

Lemma 3. *Let $a \in bh(b)$. Let k be the smallest index such that $fr(k, b+1) = hfr(b+1)$. If $a < k$, then $a \in bh(b+1)$.*

Proof. Assume that $a \notin bh(b+1)$, that is, there are i and j such that $1 \leq i < a \leq j \leq b+1$ such that $fr(i, a-1) \geq fr(a, j)$. We must have $j = b+1$. Otherwise $a \notin bh(b)$. Note that $fr(a, b+1) < fr(k, b+1)$, which implies $fr(a, k-1) < fr(a, b+1)$. Since $k - 1 \leq b$, we have $a \notin bh(b)$, which is a contradiction. □

These lemmata give us a simple approach. Start from $(a_1, \ldots, a_K, b+1)$ and delete entries until you find index k such that $fr(k, b+1)$ is maximal. We will see later in Proposition 3 that we can easily check the maximality.

Unfortunately, as demonstrated in [3] there can be $\Theta(b^{2/3})$ entries in $bh(b)$. Hence, checking every pair will not quite yield a linear algorithm. In order to achieve linearity, we use two additional bounds. Consider Figure 3(a). We have $bh(a_5 - 1) = (a_1, a_2, a_4)$. First, since $tfr(a_5) = fr(a_5, a_6 - 1) > fr(a_1, a_2 - 1)$, we have $a_5 - 1 \notin ct(a_1)$. Consequently, we do not need to check the pair $(a_1, a_5 - 1)$. Secondly, we know that for any $k \geq a_5$ we have $fr(a_5, k) \leq tfr(a_5) \leq fr(a_3, a_4 - 1)$. Hence, $a_4 \notin ch(k)$ and we can ignore a_4 after we have checked $(a_4, a_5 - 1)$. We can now put these ideas together in a single algorithm, given as Algorithm 2, and which we will refer to as the SCAN algorithm.

Proposition 2 stated that it is enough to consider intervals where then end points are each other candidates. The next proposition shows that SCAN actually tests all such pairs. Consequently, we are guaranteed to find the optimal solution.

Proposition 3. *Let p and n be count vectors and let o and z be two integers.* SCAN(p, n, o, z) *tests every pair (a, b) where $a \in ch(b)$ and $b \in ct(a)$.*

To show this, we first need the following lemma.

Lemma 4. *Let* $(a_1, \ldots, a_L) = bh(b)$. *Then* $fr(a_{k-1}, a_k - 1) < fr(a_k, a_{k+1} - 1)$.

Proof. Assume that $fr(a_{k-1}, a_k - 1) \geq fr(a_k, a_{k+1} - 1)$. Then $a_k \notin bh(b)$. □

By which we can proceed with the proof for Proposition 3.

Proof. Let us first prove that B at bth step is equal to $bh(b)$. We prove this using induction. The case $b = 1$ is trivial and assume that the result hold for $b - 1$. Let $(a_1, \ldots, a_L) = bh(b-1)$. Lemma 1 implies that $bh(b) \subseteq (a_1, \ldots, a_L, b)$.

Assume that $fr(b, b) > fr(a_L, i - 1) = hfr(b - 1)$. By definition, $b \in bh(b)$. Lemma 2 implies that b is the smallest index k for which $fr(k, b) = hfr(b)$. Lemma 3 now states that $bh(b) = (a_1, \ldots, a_L, b)$ which is exactly what we get since the while loop on Line 5 is not executed.

Assume that $fr(b, b) \leq fr(a_L, i - 1)$. Then $b \notin bh(b)$ and indeed it is deleted in the first run of the while loop (Line 5). Let $a_k \in bh(b)$ be the first entry in B after the while loop has finished. Let $a_l \in bh(b)$ be the smallest index for which $fr(a_l, b) = hfr(b)$. We claim that $k = l$. If $l > k$, then $fr(a_l, b) \leq fr(a_{l-1}, a_l - 1)$ which implies that $fr(a_l, b) \leq fr(a_{l-1}, b)$ which is a contradiction. Assume that $l < k$. By definition of k, we have $fr(a_{k-1}, a_k - 1) < fr(a_k, b)$. Lemma 4 implies that $fr(a_l, a_k - 1) \leq fr(a_{k-1}, a_k - 1)$. Hence, $fr(a_l, b) < fr(a_k, b)$, which is a contradiction. Consequently, $k = l$. Lemma 3 now states that $bh(b) = (a_1, \ldots, a_k)$ which is exactly what we have.

Now that we have proved that B at bth step is equal to $bh(b)$. Let us consider the list C. Let $a \in B \setminus C$. This means that a was deleted during some previous round, say $k < i$, and that there is j such that $fr(j, a - 1) \geq tfr(k + 1) \geq fr(k + 1, b)$. Hence a is not a head candidate of b. Consequently, all head candidates of b are included in C at bth step.

Not all entries of C are tested during the bth step. Assume that we have completed bth step and C_k is not tested ($k > 1$). Since C is a subset of the border list, Lemma 4 implies that $fr(C_k, C_{k-1} - 1) \leq fr(C_2, C_1 - 1) < tfr(b + 1)$. There is j such that $tfr(b + 1) = fr(b + 1, j)$. This implies that b is not a tail candidate for C_k, which completes the proof. □

Let us finish this section by demonstrating the linear execution time $\Theta(|p|)$ of Scan. To this end, note that we have three while-loops in the algorithm: two inner and one outer. During each iteration of the first inner loop we delete an entry from B, a unique number between 1 and $|p|$. Consequently, the *total* number of times we execute the first inner loop is $|p|$, at maximum. Similarly, for the second inner loop. The outer loop is executed $|p|$ times. Next, note that there are two non-trivial subroutines in the algorithm. First, on Lines 6 and 9, we need to compute frequencies. This can be done in constant time by, e.g. precomputing $c_j = \sum_{i=1}^{j} p_i$, and then using the identity $cnt(i, j; p) = c_j - c_{i-1}$. Secondly, on Line 9, we need to compute $tfr(b)$. We can precompute this in linear time by using the algorithm given in [3], which involves computing tail borders (equivalent to computing B in Scan) and applying Lemma 2. This shows that the total execution time for the algorithm is $\Theta(|p|)$.

Algorithm 3: FINDTILE$(X, \mathcal{T}, D)$

input : parent tile $X = (s, e) \times (x, y)$, current tile tree $\mathcal{T}$, dataset D
output : B, a tile optimizing $\mathcal{T} + X \to B$, see Problem 2

```
1  o ← p(X; T); z ← n(X; T); B ← X;
2  foreach c and d such that x ≤ c ≤ d ≤ y do
3      update p and n;
4      (a, b) ← SCAN(p, n, o, z);
5      Y ← (a + s − 1, b + s − 1) × (c, d);
6      if L(T + X → Y, D) < L(T + X → B, D) then B ← Y;
7      (a, b) ← SCAN(n, p, z, o);
8      Y ← (a + s − 1, b + s − 1) × (c, d);
9      if L(T + X → Y, D) < L(T + X → B, D) then B ← Y;
10 return B;
```

Now that we have a linear algorithm for discovering an optimal tile given a fixed set of columns, we need an algorithm for discovering the columns themselves. We employ a simple quadratic enumeration given in Algorithm 3. Note that SCAN assumes that the optimal tile is more dense than the background tile, we have to call SCAN twice, once normally to find the optimal dense subtile, and once with ones and zeroes reversed to find the optimal sparse subtile.

The computational complexity of FINDTILE is $\Theta(N^2 M)$ where N is the number of columns and M is the number of rows in the parent tile. However, if M is smaller than N, we can transpose the parent tile and obtain a $\Theta(NM \min(N, M))$ execution time.

5 Related Work

Frequent itemset mining [1] is perhaps the most well-known example of pattern mining. Here, however, we are not just interested in the itemsets, but also explicitly want to know which rows they cover. Moreover, we are not interested in finding all tiles, but aim to find tilings that describe the data well.

Mining sets of patterns that describe a dataset is an actively studied topic [2, 20,21]. Related in that it employs MDL, is the KRIMP algorithm [21], which proposed the use of MDL to identify pattern sets. Geerts et al. discuss mining large tiles of only 1s [7]. Different from these approaches, our models are hierarchical, and do allow for noise within tiles.

Kontonasios and De Bie discuss ranking a candidate collection of tiles, employing a maximum entropy model of the data to measure the interestingness of a tile [5,12]. Boolean matrix factorisation [15] can be regarded as a tile mining. The goal is to find a set of Boolean factors such that the Boolean product thereof (essentially tiles of only 1s) approximates the dataset with little error. Similarly, bi-clustering can be regarded as a form of tiling, as it partitions the rows and columns of a dataset into rectangles [18]. Compared to these approaches,

a major difference is that we focus on easily inspectable hierarchical models, allowing nested refinements within tiles.

Most closely related to Stijl is the approach by Gionis et al. [8], who proposed mining hierarchies of tiles, and gave a randomised heuristic for finding good subtiles. We improve over this approach by formally defining the problem in terms of MDL, employing a richer modelling language in the sense that it allows tiles with the same parent to overlap, introducing a deterministic iterative any-time algorithm, that given a tile tree efficiently finds the optimal subtile. Since the approach by Gionis et al. [8] does not use MDL as a stopping criterion, it is not possible to compare both methods directly. In principle, it is possible to adopt their search strategy to our score. A fair comparison between the two search strategies, however, is not trivial since the randomised search depends on a parameter, namely the number of restarts. This parameter acts as a trade-off between the expected performance and execution time. Choosing this parameter is difficult since there are no known bounds for the expected performance.

Our approach for discovering optimal subtiles is greatly inspired by the work of Calders et al. [3] in which the goal was to compute the head frequency, $hfr(i)$, given a stream of binary vectors.

As Stijl is an iterative any-time algorithm, and hence iterative data mining approaches are related. The key idea of these approaches is to iteratively find the result providing the most novel information about the data with respect to what we already know [5,11,14]. Here, we focus on hierarchical tiles, and efficiently find the locally optimal addition.

6 Experiments

In this section we empirically evaluate our approach. We implemented our algorithms in C++, and provide the source code, along with the synthetic data generator[3]. All experiments were executed single-threaded on Linux machines with Intel Xeon X5650 processors (2.66GHz) and 12 GB of memory.

We use the shorthand notation $L\%$ to denote the compressed size of D with the tile tree $\mathcal{T}$ as discovered by Stijl relative to the most simple tree $\mathcal{T}_0$, $\frac{L(D,\mathcal{T})}{L(D,\mathcal{T}_0)}\%$, wherever D and $\mathcal{T}$ are clear from context.

We do not compare to the naive strategy of finding optimal subtiles as $\Theta(N^2M^2)$ execution time is impractical even for very small datasets.

Datasets. We evaluate our measure on one synthetic, and four publicly available real world datasets. The 240-by-240 synthetic dataset *Composition* was generated to the likeness of the famous Mondrian painting 'Composition II in Red, Blue, and Yellow', where we use different frequencies of 1s for each of the colours. *Abstracts* contains the abstracts of papers accepted at ICDM up to 2007, for which we take the words with a frequency of at least 0.02 after stemming and removing stop words [5]. The *DNA* amplification data contains DNA copy number amplifications. Such copies are known to activate oncogenes and are

[3] http://adrem.ua.ac.be/stijl/

Table 1. Results of Stijl on five datasets. Shown are, per dataset, number of rows and columns, overall density, and for resp. without and with overlap, the relative compression $L\%$ (lower is better), number of discovered tiles, and wall-clock runtime.

				Disjoint			**Overlap**		
Dataset	N	M	%1s	$L\%$	$\lvert\mathcal{T}\rvert$	*time*	$L\%$	$\lvert\mathcal{T}\rvert$	*time*
Composition	240	240	23.2	81.72	8	57s	81.58	7	1m23s
Abstracts	859	541	6.6	89.59	14	16m03s	89.54	14	27m54s
DNA Amp.	4 590	391	1.5	61.91	466	334m	61.61	446	625m
Mammals	2 183	121	20.5	54.69	55	1m37s	54.62	50	3m06s
Paleo	501	139	5.1	80.23	14	39s	79.07	13	1m22s

the hallmarks of nearly all advanced tumours [17]. The *Mammals* presence data consists of presence records of European mammals[4] within geographical areas of 50×50 kilometers [16]. Finally, *Paleo* contains information on fossil records[5] found at specific palaeontological sites in Europe [6].

We give the basic properties of these datasets in Table 1. To obtain good orders for the real world datasets, we applied SVD, that is, we ordered items and transactions based on first left and right eigenvectors.

Synthetic Data. As a sanity check, we first investigate whether Stijl can reconstruct the model for the *Composition* data. We ran experiments for both the disjoint and the overlapping tile settings. With the latter setup we perfectly capture the underlying model in only 7 tiles. We show the data and discovered model as Figure 4(a); each of the rectangles in the painting are represented correctly by a tile, including the crossing vertical and horizontal bars. When we require disjoint tiles, the fit of the model is equally good, however the model requires one additional tile to model the crossing bars.

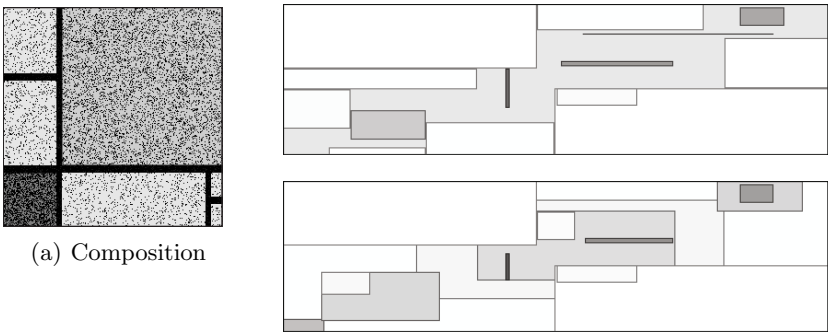
Quantitative Analysis. Next, we consider quantitative results on the real datasets. We run Stijl both for disjoint and overlapping tiles. Table 1 gives the relative compressed size $L\%$, the number of tiles in the returned tile trees, and the wall-clock time it took to find these models.

These results show Stijl finds trees that summarise the data well. The relative compressed sizes tell the data is described succinctly, while the tile trees remain small enough to be considered by hand; even for *DNA*, by the hierarchical nature of the model, the user can quickly read and understand the model.

For *Mammals* we see that whereas the baseline model $\mathcal{T}_0$ requires 193 315 bits, the Stijl model with overlapping tiles only requires 105 589 bits. Note that, as the total compressed size essentially is the log-likelihood of the model and the data, a gain of a single bit corresponds to a twice as likely model.

In all experiments, allowing overlap results in better models. Not only do they give more succinct data descriptions, the discovered tile trees are also simpler,

[4] Available for research purposes: `http://www.european-mammals.org`
[5] NOW public release 030717 available from [6].



(a) Composition

(b) Paleo (transposed)

Fig. 4. Results of Stijl on (a) *Composition* and (b) *Paleo*, with (top) the disjoint hierarchical tiling, and (bottom) the tiling allowing overlap within the same parent tile. For *Paleo* we do not show individual 1s. Darker tiles correspond to higher frequency.

requiring fewer tiles to do capture the structure of the data. By allowing overlap, the search space is expanded, and hence more computation is required: on average, in our experiments, twice as much.

On these datasets, the current Stijl implementation requires from seconds up to a few hours of runtime. By its iterative any-time nature, users, however, can already start to explore models while in the background further refinements are calculated.

Qualitative Analysis. Next, we investigate the discovered models in more detail. To this end, we first use the *Paleo* data as by its modest size it is easily visually representable. In Figure 4(b) we show the result of Stijl on this data, with the top figure the result of allowing only disjoint tiles, and in the bottom figure when allowing overlap. Darker toned tiles correspond to more dense areas of the data. For clarity, we here do not show the individual 1s (as we did in Fig. 2, which corresponds to the bottom plot of Fig. 4(b)).

The first thing we note, is that the two results are quite alike. The model with overlap, however, is a bit simpler and 'cleaner': the relatively dense areas are of the data are easier to spot for this model, than for the disjoint one. Second, it uses the hierarchical property as intended: in the top right corner, for instance, we see a dense, dark-grey tile within a lighter tinted square, within a very sparse tile. While for reasons of space we can only show these examples, these are observations that hold in general—by which it may come at no surprise that by allowing overlap we obtain better MDL scores.

Next, we inspect the results on *Abstracts*. This sparse dataset has no natural order by itself, and when we apply SVD to order it, we find most of the 1s are located in the top-left corner of the data. When we apply Stijl, we see it correctly reconstructs this structure. Due to lack of space, however, we do not give the visual representation. Instead, we investigate the most dense tile, which covers the top-left corner. We find that it includes frequent words that are

often used in conjunction in data mining abstracts, including *propose*, *efficient*, *method*, *mine*, and *algorithm*. Note that, by design, StijL gives a high level view of the data; that is, it tells you where the ones are, not necessarily their associations. Extending it to recognise structure within tiles is future work.

7 Discussion

The experiments show StijL discovers succinct tile trees that summarise the data well. Importantly, the discovered tile trees consist of only few tiles, and are even easier inspected by the hierarchical property of our models.

The complexity of StijL is much lower than that of the naive locally optimal approach; as with $\Theta(NM \min(N, M))$ its complexity is only squared in the smallest dimension of the data. However, for datasets with both many rows and columns, runtimes may be non-trivial. StijL, however, does allow ample opportunity for optimisation. FindTile, for instance, can be trivially run in parallel per parent tile, as well as over a and b.

As there is no such thing as a free lunch, we have to note that MDL is no magic wand. In the end, constructing an encoding involves choices—choices one can make in a principled manner (fewer bits is better), but choices nevertheless. Here, our choices were bounded by ensuring optimality of FindTile. As such, we currently ignore globally optimal encoding solutions, such as achievable by maximum entropy modelling [5]. Although we could so obtain globally optimality of the encoding, the effects of adding a tile become highly unpredictable, which would break the locally optimal search of FindTile.

We assume the rows and columns of the data to be ordered. That is, in the terminology of [8], we are interested in geometric tiles. Although [6,8] showed good geometric tilings can be found on spectrally ordered data, it would make for engaging research to investigate whether we can find good orderings on the fly, that is while we are tiling, ordering the data such that we optimise our score.

8 Conclusion

We discussed finding good hierarchical tile-based models for binary data. We formalised the problem in terms of MDL, and introduced the StijL algorithm for greedily approximating the score on binary data with ordered rows and columns. For unordered data, spectral techniques can be used to find good orders [8]. We gave the FindTile procedure for which we proved it finds the locally optimal tile in $\Theta(NM \min(N, M))$.

Experiments showed StijL discovers high-quality tile trees, providing succinct description of binary data. Importantly, by their hierarchical shape and small size, these models are easily interpreted and analysed by hand.

Future work includes optimising the encoded cost by mining tiles and orders at the same time, as opposed to using ordering techniques oblivious to the target.

Acknowledgements. Nikolaj Tatti and Jilles Vreeken are supported by Post-Doctoral Fellowships of the Research Foundation – Flanders (FWO).

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB, pp. 487–499 (1994)
2. Bringmann, B., Zimmermann, A.: The chosen few: On identifying valuable patterns. In: ICDM, pp. 63–72 (2007)
3. Calders, T., Dexters, N., Goethals, B.: Mining frequent itemsets in a stream. In: ICDM, pp. 83–92. IEEE (2007)
4. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience, New York (2006)
5. De Bie, T.: Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Min. Knowl. Disc. 23(3), 407–446 (2011)
6. Fortelius, M., Gionis, A., Jernvall, J., Mannila, H.: Spectral ordering and biochronology of european fossil mammals. Paleobiology 32(2), 206–214 (2006)
7. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling Databases. In: Suzuki, E., Arikawa, S. (eds.) DS 2004. LNCS (LNAI), vol. 3245, pp. 278–289. Springer, Heidelberg (2004)
8. Gionis, A., Mannila, H., Seppänen, J.K.: Geometric and Combinatorial Tiles in 0–1 Data. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 173–184. Springer, Heidelberg (2004)
9. Grünwald, P.: The Minimum Description Length Principle. MIT Press (2007)
10. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: Current status and future directions. Data Min. Knowl. Disc. 15 (2007)
11. Hanhijärvi, S., Ojala, M., Vuokko, N., Puolamäki, K., Tatti, N., Mannila, H.: Tell me something I don't know: randomization strategies for iterative data mining. In: KDD, pp. 379–388. ACM (2009)
12. Kontonasios, K.-N., De Bie, T.: An information-theoretic approach to finding noisy tiles in binary databases. In: SDM, pp. 153–164. SIAM (2010)
13. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and its Applications. Springer (1993)
14. Mampaey, M., Tatti, N., Vreeken, J.: Tell me what I need to know: Succinctly summarizing data with itemsets. In: KDD, pp. 573–581. ACM (2011)
15. Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. IEEE TKDE 20(10), 1348–1362 (2008)
16. Mitchell-Jones, A., Amori, G., Bogdanowicz, W., Krystufek, B., Reijnders, P.H., Spitzenberger, F., Stubbe, M., Thissen, J., Vohralik, V., Zima, J.: The Atlas of European Mammals. Academic Press (1999)
17. Myllykangas, S., Himberg, J., Böhling, T., Nagy, B., Hollmén, J., Knuutila, S.: DNA copy number amplification profiling of human neoplasms. Oncogene 25(55), 7324–7332 (2006)
18. Pensa, R.G., Robardet, C., Boulicaut, J.-F.: A Bi-clustering Framework for Categorical Data. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 643–650. Springer, Heidelberg (2005)
19. Tatti, N.: Are your items in order? In: SDM 2011, pp. 414–425. SIAM (2011)
20. Tatti, N., Heikinheimo, H.: Decomposable Families of Itemsets. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 472–487. Springer, Heidelberg (2008)
21. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: Mining itemsets that compress. Data Min. Knowl. Disc. 23(1), 169–214 (2011)

Efficient Discovery of Association Rules and Frequent Itemsets through Sampling with Tight Performance Guarantees*

Matteo Riondato and Eli Upfal

Department of Computer Science, Brown University, Providence, RI, USA
{matteo,eli}@cs.brown.edu

Abstract. The tasks of extracting (top-K) Frequent Itemsets (FI's) and Association Rules (AR's) are fundamental primitives in data mining and database applications. Exact algorithms for these problems exist and are widely used, but their running time is hindered by the need of scanning the entire dataset, possibly multiple times. High quality approximations of FI's and AR's are sufficient for most practical uses, and a number of recent works explored the application of sampling for fast discovery of approximate solutions to the problems. However, these works do not provide satisfactory performance guarantees on the quality of the approximation, due to the difficulty of bounding the probability of under- or over-sampling any one of an unknown number of frequent itemsets. In this work we circumvent this issue by applying the statistical concept of *Vapnik-Chervonenkis (VC) dimension* to develop a novel technique for providing tight bounds on the sample size that guarantees approximation within user-specified parameters. Our technique applies both to absolute and to relative approximations of (top-K) FI's and AR's. The resulting sample size is linearly dependent on the VC-dimension of a range space associated with the dataset to be mined. The main theoretical contribution of this work is a characterization of the VC-dimension of this range space and a proof that it is upper bounded by an easy-to-compute characteristic quantity of the dataset which we call *d-index*, namely the maximum integer d such that the dataset contains at least d transactions of length at least d. We show that this bound is strict for a large class of datasets. The resulting sample size for an absolute (resp. relative) (ε, δ)-approximation of the collection of FI's is $O(\frac{1}{\varepsilon^2}(d + \log \frac{1}{\delta}))$ (resp. $O(\frac{2+\varepsilon}{\varepsilon^2(2-\varepsilon)\theta}(d \log \frac{2+\varepsilon}{(2-\varepsilon)\theta} + \log \frac{1}{\delta}))$) transactions, which is a significant improvement over previous known results. We present an extensive experimental evaluation of our technique on real and artificial datasets, demonstrating the practicality of our methods, and showing that they achieve even higher quality approximations than what is guaranteed by the analysis.

1 Introduction

Discovery of frequent itemsets and association rules is a fundamental computational primitive with application in data mining (market basket analysis), databases (histogram construction), networking (heavy hitters) and more [15, Sect. 5]. Depending on the particular application, one is interested in finding all itemsets with frequency greater or

* Work was supported in part by NSF award IIS-0905553.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 25–41, 2012.

equal to a user defined threshold (FIs), identifying the K most frequent itemsets (top-K), or computing all association rules (ARs) with user defined minimum support and confidence level. Exact solutions to these problems require scanning the entire dataset, possibly multiple times. For large datasets that do not fit in main memory, this can be prohibitively expensive. Furthermore, such extensive computation is often unnecessary, since high quality approximation are sufficient for most practical applications. Indeed, a number of recent papers [4, 6, 7, 9, 10, 12, 13, 17–22, 25, 27–30, 32, 33, 36–41] explored the application of sampling for approximate solutions to these problems. However, the efficiency and practicality of the sampling approach depends on a tight relation between the size of the sample and the quality of the resulting approximation. Previous works do not provide satisfactory solutions to this problem.

The technical difficulty in analyzing any sampling technique for frequent itemsets discovery problems is that a-priori any subset of items can be among the most frequent ones, and the number of subsets is exponential in the number of distinct items appearing in the dataset. A standard analysis begins with a bound on the probability that a given itemset is either over or under represented in the sample. Such bound is easy to obtain using a Chernoff-like bound or the Central Limit theorem. The difficulty is in combining the bounds for individual itemsets into a global bound that holds simultaneously for all the itemsets. A simple application of the union bound vastly overestimates the error probability because of the large number of possible itemsets, a large fraction of which may not be present in the dataset and therefore should not be considered. More sophisticated techniques, developed in recent works [6, 12, 29], give better bounds only in limited cases. A loose bound on the required sample size for achieving the user defined performance guarantees, decreases the gain obtained from the use of sampling.

In this work we circumvent this problem through a novel application of the *Vapnik-Chervonenkis (VC)* dimension concept, a fundamental tool in statistical learning theory. Roughly speaking, the VC-dimension of a collection of indicator functions (a range space) is a measure of its complexity or expressiveness (see Sect. 2.2 for formal definitions). A major result [35] relates the VC-dimension of a range space to the sufficient size for a random sample to simultaneously approximate all the indicator functions within predefined parameters. The main obstacle in applying the VC-dimension theory to particular computation problems is computing the VC-dimension of the range spaces associated with these problems.

We apply the VC-dimension theory to frequent itemsets problems by viewing the presence of an itemset in a transaction as the outcome of an indicator function associated with the itemset. The major theoretical contributions of our work are a complete caractherization of the VC-dimension of the range space associated with a dataset, and a tight bound to this quantity. We prove that the VC-dimension is upper bounded by an easy-to-compute characteristic quantity of the dataset which we call *d-index*, namely, the maximum integer d such that the dataset contains at least d transactions of length at least d. We show that this bound is tight by demonstrating a large class of datasets with a VC-dimension that matches the bound.

The VC-dimension approach provides a unified tool for analyzing the various frequent itemsets and association rules problems (i.e., the market basket analysis tasks). We use it to prove tight bounds on the required sample size for extracting FI's with a

minimum frequency threshold, for mining the top-K FI's, and for computing the collection of AR's with minimum frequency and confidence thresholds. Furthermore, we compute bounds for both absolute and relative approximations (see Sec 2.1 for definitions). We show that high quality approximations can be obtained by mining a very small random sample of the dataset. For example, the required sample size for an absolute (ε, δ)-approximation of the collection of FI's is $O(\frac{1}{\varepsilon^2}(d + \log \frac{1}{\delta}))$ transactions, which is a significant improvement over previous known results, as it is smaller and, more importantly, less dependent on parameters such as the minimum frequency threshold and the dataset size. Similar results are proven for the top-K FI's and AR's tasks.

We present an extensive experimental evaluation of our method using real and artificial datasets, to assess the practicality of our approach. The experimental results show that indeed our method achieves, and even exceeds, the analytically proven guarantees for the quality of the approximations.

1.1 Previous Work

Agrawal et al. [1] introduced the problem of mining association rules in the basket data model, formalizing a fundamental task of information extraction in large datasets. Almost any known algorithm for the problem starts by solving a FI's problem and then generate the association rules implied by these frequent itemsets. Agrawal and Srikant [2] presented *Apriori*, the most well-known algorithm for mining FI's, and *FastGenRules* for computing association rules from a set of itemsets. Various ideas for improving the efficiency of FI's and AR's algorithms have been studied, and we refer the reader to the survey by Ceglar and Roddick [5] for a good presentation of recent contributions. However, the running times of all known algorithms heavily depend on the size of the dataset.

Mannila et al. [27] first suggested the idea that sampling can be used to efficiently obtain the collection of FI's, presenting some empirical results to validate the intuition. Toivonen [33] presents an algorithm that, by mining a random sample of the dataset, builds a candidate set of frequent itemsets which contains all the frequent itemsets with a probability that depends on the sample size. There are no guarantees that that all itemsets in the candidate set are frequent, but the set of candidates can be used to efficiently identify the set of frequent itemsets with at most two passes over the entire dataset. The work also suggests a bound on the sample size sufficient to ensure that the frequencies of itemsets in the sample are close to their real one. The analysis uses Chernoff bounds and the union bound. The major drawback of this sample size is that it depends linearly on the number of individual items appearing in the dataset.

Zaki et al. [39] show that static sampling is an efficient way to mine a dataset, but choosing the sample size using Chernoff bounds is too conservative, in the sense that it is possible to obtain the same accuracy and confidence in the approximate results at smaller sizes than what the theoretical analysis suggested.

Other works tried to improve the bound to the sample size by using different techniques from statistic and probability theory like the central limit theorem [19, 22, 40] or hybrid Chernoff bounds [41].

Since theoretically-derived bounds to the sample size where too loose to be useful, a corpus of works applied progressive sampling to extract FI's [4, 7, 9, 10, 12, 17, 18, 20,

21, 25, 28, 38]. Progressive sampling algorithms work by selecting a random sample and then trimming or enriching it by removing or adding new sampled transactions according to a heuristic or a self-similarity measure that is fast to evaluate, until a suitable stopping condition is satisfied. The major downside of this approach is that it offers no guarantees on the quality of the obtained results.

Another approach to estimating the required sample size is presented in [13]. The authors give an algorithm that studies the distribution of frequencies of the itemsets and uses this information to fix a sample size for mining frequent itemsets, but without offering any theoretical guarantee.

A recent work by Chakaravarthy et al. [6] gives the first analytical bound on a sample size that is linear in the length of the longest transaction, rather than in the number of items in the dataset. This work is also the first to present an algorithm that uses a random sample of the dataset to mine approximated solutions to the AR's problem with quality guarantees. No experimental evaluation of their methods is presented, and they do not address the top-K FI's problem. Our approach gives better bounds for the problems studied in [6] and applies to related problems such as the discovery of top-K FI's and absolute approximations.

Extracting the collection of top-K frequent itemsets is a more difficult task since the corresponding minimum frequency threshold is not known in advance [11, 14]. Some works solved the problem by looking at *closed* top-K frequent itemsets, a concise representation of the collection [30, 37], but they suffers from the same scalability problems as the algorithms for exactly mining FI's with a fixed minimum frequency threshold.

Previous works that used sampling to approximation the collection of top-K FI's [29, 32] used progressive sampling. Both works provide (similar) theoretical guarantees on the quality of the approximation. What is more interesting to us, both works present a theoretical upper bound to the sample size needed to compute such an approximation. The size depended linearly on the number of items. In contrast, our results give a sample size that only in the worst case is linear in the number of items but can be (and is, in practical cases) much less than that, depending on the dataset, a flexibility not provided by previous contributions. Sampling is used by Vasudevan and Vojonović [36] to extract an approximation of the top-K frequent individual *items* from a sequence of items, which contains no item whose actual frequency is less than $f_K - \varepsilon$ for a fixed $0 < \varepsilon < 1$, where f_K is the *actual* frequency of the K-th most frequent item. They derive a sample size sufficient to achieve this result, but they assume the knowledge of f_K, which is rarely the case. An empirical sequential method can be used to estimate the right sample size. Moreover, the results cannot be directly extended to the mining of top-K frequent item(set)s from datasets of transactions with length greater than one.

1.2 Our Contributions

By applying tools from statistical learning theory, we develop a general technique for bounding the sample size required for generating high quality approximations to frequent itemsets and association rules tasks. Table 1 compares our technique to the best previously known results for the various problems (see Sect. 2.1 for definitions). Our bounds, which are linear in the VC-dimension associated with the dataset, are consistently smaller and less dependent on other parameters of the problem than

Table 1. Required sample sizes (as number of transactions) as a function of the VC-dimension d, the maximum transaction size Δ, the number of items $|\mathcal{I}|$, the accuracy ε, the failure probability δ, the minimum frequency θ, and the minimum confidence γ. Note that $d \leq \Delta \leq |\mathcal{I}|$.

Task	Approx.	This work	Best previous work
FI's	absolute	$\frac{4c}{\varepsilon^2}\left(d + \log\frac{1}{\delta}\right)$	$O\left(\frac{1}{\varepsilon^2}\left(\lvert\mathcal{I}\rvert + \log\frac{1}{\delta}\right)\right)$ [19, 22, 33, 40]
	relative	$\frac{4(2+\varepsilon)c}{\varepsilon^2(2-\varepsilon)\theta}\left(d\log\frac{2+\varepsilon}{\theta(2-\varepsilon)} + \log\frac{1}{\delta}\right)$	$\frac{24}{\varepsilon^2(1-\varepsilon)\theta}\left(\Delta + 5 + \log\frac{4}{(1-\varepsilon)\theta\delta}\right)$ [6]
top-K	absolute	$\frac{16c}{\varepsilon^2}\left(d + \log\frac{1}{\delta}\right)$	$O\left(\frac{1}{\varepsilon^2}\left(\lvert\mathcal{I}\rvert + \log\frac{1}{\delta}\right)\right)$ [29, 32]
	relative	$\frac{4(2+\varepsilon)c}{\varepsilon^2(2-\varepsilon)\theta}\left(d\log\frac{2+\varepsilon}{\theta(2-\varepsilon)} + \log\frac{1}{\delta}\right)$	not available
AR's	absolute	$O\left(\frac{(1+\varepsilon)}{\varepsilon^2(1-\varepsilon)\theta}\left(d\log\frac{1+\varepsilon}{\theta(1-\varepsilon)} + \log\frac{1}{\delta}\right)\right)$	not available
	relative	$\frac{16c(4+\varepsilon)}{\varepsilon^2(4-\varepsilon)\theta}\left(d\log\frac{4+\varepsilon}{\theta(4-\varepsilon)} + \log\frac{1}{\delta}\right)$	$\frac{48}{\varepsilon^2(1-\varepsilon)\theta}\left(\Delta + 5 + \log\frac{4}{(1-\varepsilon)\theta\delta}\right)$ [6]

previous results. An extensive experimental evaluation demonstrates the advantage of our technique in practice.

To the best of our knowledge, our work is the first to provide a caractherization and an explicit bound for the VC-dimension of the range space associated to a dataset and to apply the result to the extraction of FI's and AR's from random sample of the dataset. We believe that this connection with statistical learning theory can be furtherly exploited in other data mining problems.

We also believe that our approach can be applied not just to mining collections of frequent itemsets and association rules, which can be massive, but also to the mining of small collections of itemsets/association rules that describe the dataset with the minimal number of itemsets/association rules possible, as presented in [26].

Outline. In Sect. 2 we formally define the problem and our goals, and introduce definitions and lemmas used in the analysis. The main part of the analysis with derivation of a strict bound to the VC-dimension of association rules is presented in Sect. 3, while our algorithms and sample sizes for mining FI's, top-K FI's, and association rules through sampling are in Sect. 4. Section 5 contains an extensive experimental evaluation of our techniques. Due to space constraints, the proofs of our theorems and lemmas are not presented in this paper. We refer the interested reader to the full version [31].

2 Preliminaries

2.1 Datasets, Itemsets, and Association Rules

A *dataset* $\mathcal{D}$ is a collection of *transactions*, where each transaction τ is a subset of a ground set $\mathcal{I}$. There can be multiple identical transactions in $\mathcal{D}$. Members of $\mathcal{I}$ are called *items* and members of $2^{\mathcal{I}}$ are called *itemsets*. Let $|\tau|$ denote the number of items in transaction τ. Given an itemset $A \in 2^{\mathcal{I}}$, let $T_{\mathcal{D}}(A)$ denote the set of transactions in $\mathcal{D}$ that contain A. The *support* of A, $\sigma_{\mathcal{D}}(A) = |T_{\mathcal{D}}(A)|$, is the number of transaction in D that contains A, and the *frequency* of A, $f_{\mathcal{D}}(A) = \frac{|T_{\mathcal{D}}(A)|}{|\mathcal{D}|}$, is the fraction of transactions in $\mathcal{D}$ that contain A.

Definition 1. *Given a* minimum frequency threshold θ, $0 < \theta \leq 1$, *the* FI's mining task with respect to θ *is finding all itemsets with frequency* $\geq \theta$, *i.e., the set*

$$\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta) = \{(A, f_{\mathcal{D}}(A)) \ : \ A \in 2^{\mathcal{I}} \text{ and } f_{\mathcal{D}}(A) \geq \theta\}.$$

To define the collection of top-K FI's, we assume a fixed *canonical ordering* of the itemsets in $2^{\mathcal{I}}$ by decreasing frequency in $\mathcal{D}$, with ties broken arbitrarily, and label the itemsets $A_1, A_2, \ldots, A_m$ according to this ordering. For a given K, with $1 \leq K \leq m$, we denote with $f_{\mathcal{D}}^{(K)}$ the frequency $f_{\mathcal{D}}(A_K)$ of the K-th most frequent itemset A_K, and define the set of top-K FI's (with their respective frequencies) as

$$\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K) = \mathsf{FI}(\mathcal{D}, \mathcal{I}, f_{\mathcal{D}}^{(K)}).$$

One of the main uses of frequent itemsets is in the discovery of *association rules*.

Definition 2. *An* association rule W *is an expression "$A \Rightarrow B$" where A and B are itemsets such that $A \cap B = \emptyset$. The* support $\sigma_{\mathcal{D}}(W)$ *(resp. frequency $f_{\mathcal{D}}(W)$) of the association rule W is the support (resp. frequency) of the itemset $A \cup B$. The* confidence $c_{\mathcal{D}}(W)$ *of W is the ratio $\frac{f_{\mathcal{D}}(A \cup B)}{f_{\mathcal{D}}(A)}$ of the frequency of $A \cup B$ to the frequency of A.*

Intuitively, an association rule "$A \Rightarrow B$" expresses, throught its support and confidence, how likely it is for the itemset B to appear in the same transactions as itemset A, so that when A is found in a transaction it is then possible to infer that B will be present in the same transaction with a probability equal to the confidence of the association rule.

Definition 3. *Given a dataset $\mathcal{D}$ with transactions built on a ground set $\mathcal{I}$, and given a minimum frequency threshold θ and a minimum confidence threshold γ, the* AR's task with respect to θ and γ *consist in finding the set*

$$\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma) = \{(W, f_{\mathcal{D}}(W), c_{\mathcal{D}}(W)) \mid \textit{Assoc. Rule } W, f_{\mathcal{D}}(W) \geq \theta, c_{\mathcal{D}}(W) \geq \gamma\}.$$

Often, with an abuse of the notation, we will say that an itemset A (resp. an association rule W) is in $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ or in $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$ (resp. in $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$) and denote this fact with $A \in \mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ or $A \in \mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$ (resp. $W \in \mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$), meaning that there is a pair $(A, f) \in \mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ or $(A, f) \in \mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$ (resp. a triplet $(W, f_w, c_w) \in \mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$).

In this work we are interested in extracting absolute and relative approximations of the sets $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$ and $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$.

Definition 4. *Given a parameter ε_{abs} (resp. ε_{rel}), an* absolute ε_{abs}-close approximation *(resp. a* relative ε_{rel}-close approximation*) of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ is a set $\mathcal{C} = \{(A, f_A) \ : \ A \in 2^{\mathcal{I}}, f_A \in [0, 1]\}$ of pairs (A, f_A) where f_A approximates $f_{\mathcal{D}}(A)$. $\mathcal{C}$ is such that:*

1. *$\mathcal{C}$ contains all itemsets appearing in $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$;*
2. *$\mathcal{C}$ contains no itemset A with frequency $f_{\mathcal{D}}(A) < \theta - \varepsilon_{\text{abs}}$ (resp. $f_{\mathcal{D}}(A) < (1 - \varepsilon_{\text{rel}})\theta$);*
3. *For every pair $(A, f_A) \in \mathcal{C}$, it holds $|f_{\mathcal{D}}(A) - f_A| \leq \varepsilon_{\text{abs}}$ (resp. $|f_{\mathcal{D}}(A) - f_A| \leq \varepsilon_{\text{rel}} f_{\mathcal{D}}(A)$).*

This definition extends easily to the case of top-K frequent itemsets mining using the equivalence $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K) = \mathsf{FI}\left(\mathcal{D}, \mathcal{I}, f_{\mathcal{D}}^{(K)}\right)$: an absolute (resp. relative) ε-close approximation to $\mathsf{FI}\left(\mathcal{D}, \mathcal{I}, f_{\mathcal{D}}^{(K)}\right)$ is an absolute (resp. relative) ε-close approximation to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$.

For the case of association rules, we have the following definition.

Definition 5. *Given a parameter ε_{abs} (resp. ε_{rel}), an* absolute ε_{abs}-close approximation *(resp. a* relative ε_{rel}-close approximation*) of $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ is a set*

$$\mathcal{C} = \{(W, f_W, c_W) \ : \ \textit{association rule } W, f_W \in [0,1], c_W \in [0,1]\}$$

of triplets (W, f_W, c_W) where f_W and c_W approximate $f_{\mathcal{D}}(W)$ and $c_{\mathcal{D}}(W)$ respectively. $\mathcal{C}$ is such that:

1. *$\mathcal{C}$ contains all association rules appearing in $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$;*
2. *$\mathcal{C}$ contains no association rule W with frequency $f_{\mathcal{D}}(W) < \theta - \varepsilon_{\text{abs}}$ (resp. $f_{\mathcal{D}}(W) < (1 - \varepsilon_{\text{rel}})\theta$);*
3. *For every triplet $(W, f_W, c_W) \in \mathcal{C}$, it holds $|f_{\mathcal{D}}(W) - f_W| \leq \varepsilon_{\text{abs}}$ (resp. $|f_{\mathcal{D}}(W) - f_W| \leq \varepsilon_{\text{rel}}\theta$).*
4. *$\mathcal{C}$ contains no association rule W with confidence $c_{\mathcal{D}}(W) < \gamma - \varepsilon_{\text{abs}}$ (resp. $c_{\mathcal{D}}(W) < (1 - \varepsilon_{\text{rel}})\gamma$);*
5. *For every triplet $(W, f_W, c_W) \in \mathcal{C}$, it holds $|c_{\mathcal{D}}(W) - c_W| \leq \varepsilon_{\text{abs}}$ (resp. $|c_{\mathcal{D}}(W) - c_W| \leq \varepsilon_{\text{rel}} c_{\mathcal{D}}(W)$).*

Note that the definition of relative ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ (resp. to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$) is more stringent than the definition of ε-close solution to frequent itemset mining (resp. association rule mining) in [6, Sect. 3]. Specifically, we require an approximation of the frequencies (and confidences) in addition to the approximation of the collection of itemsets or association rules (property 3 in Def. 4 and properties 3 and 5 in Def. 5).

2.2 VC-Dimension

The Vapnik-Chernovenkis (VC) Dimension of a space of points is a measure of the complexity or expressiveness of a family of indicator functions (or equivalently a family of subsets) defined on that space [35]. A finite bound on the VC-dimension of a structure implies a bound on the number of random samples required for approximately learning that structure. We outline here some basic definitions and results and refer the reader to the works of Alon and Spencer [3, Sect. 14.4], Chazelle [8, Chap. 4], and Vapnik [34] for more details on VC-dimension.

VC-dimension is defined on *range spaces*:

Definition 6. *A* range space *is a pair (X, R) where X is a (finite or infinite) set and R is a (finite or infinite) family of subsets of X. The members of X are called* points *and those of R are called* ranges*.*

To define the VC-dimension of a range space we consider the projection of the ranges into a set of points:

Definition 7. *Let (X, R) be a range space and $A \subset X$. The* projection *of R on A is defined as $P_R(A) = \{r \cap A \ : \ r \in R\}$.*

The definition of *shattered* set will be heavily used in our proofs:

Definition 8. *Let (X, R) be a range space and $A \subset X$. If $P_R(A) = 2^A$, then A is said to be* shattered by *R.*

The VC-dimension of a range space is the cardinality of the largest set shattered by the space:

Definition 9. *Let $S = (X, R)$ be a range space. The* Vapnik-Chervonenkis *dimension (or* VC-dimension*) of S, denoted as $\mathsf{VC}(S)$ is the maximum cardinality of a shattered subset of X. If there are arbitrary large shattered subsets, then $\mathsf{VC}(S) = \infty$.*

The main application of VC-dimension in statistics and learning theory is its relation to the size of the sample needed to approximate learning the ranges, in the following sense.

Definition 10. *Let (X, R) be a range space and let A be a finite subset of X.*

1. *For $0 < \varepsilon < 1$, a subset $B \subset A$ is an* ε-approximation *for A if $\forall r \in R$, we have*
$$\left| \frac{|A \cap r|}{|A|} - \frac{|B \cap r|}{|B|} \right| \leq \varepsilon. \tag{1}$$
2. *For $0 < p, \varepsilon < 1$, a subset $B \subset A$ is a* relative (p, ε)-approximation *for A if for any range $r \in R$ such that $\frac{|A \cap r|}{|A|} \geq p$ we have $\left| \frac{|A \cap r|}{|A|} - \frac{|B \cap r|}{|B|} \right| \leq \varepsilon \frac{|A \cap r|}{|A|}$ and for any range $r \in R$ such that $\frac{|A \cap r|}{|A|} < p$ we have $\frac{|B \cap r|}{|B|} \leq (1 + \varepsilon)p$.*

An ε-approximation (resp. a relative (p, ε)-approximation) can be constructed by random sampling points of the point space [16, Thm. 2.12 (resp. 2.11)].

Theorem 1. *There is an absolute positive constant c (resp. c') such that if (X, R) is a range-space of VC-dimension at most d, $A \subset X$ is a finite subset and $0 < \varepsilon, \delta < 1$ (resp. and $0 < p < 1$), then a random subset $B \subset A$ of cardinality m, where*
$$m \geq \min \left\{ |A|, \frac{c}{\varepsilon^2} \left(d + \log \frac{1}{\delta} \right) \right\}, \tag{2}$$
(resp. $m \geq \min \left\{ |A|, \frac{c'}{\varepsilon^2 p} \left(d \log \frac{1}{p} + \log \frac{1}{\delta} \right) \right\}$) is an ε-approximation (resp. a relative (p, ε)-approximation) for A with probability at least $1 - \delta$.

Note that throughout the work we assume the sample to be drawn *with* replacement if $m < |A|$ (othewise the sample is exactly the set A). Löffler and Phillips [24] showed experimentally that the absolute constant c is approximately 0.5. It is also interesting to note that an ε-approximation of size $O(\frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon})$ can be built *deterministically* in time $O(d^{3d}(\frac{1}{\varepsilon^2} \log \frac{d}{\varepsilon})^d |X|)$ [8].

3 The Dataset's Range Space and Its VC-Dimension

We define one range space that is used in the derivation of the sample sizes needed to approximate the solutions to the tasks of market basket analysis.

Definition 11. *Let $\mathcal{D}$ be a dataset of transactions that are subsets of a ground set $\mathcal{I}$. We define $S = (X, R)$ to be a range space associated with $\mathcal{D}$ such that:*

1. *$X = \mathcal{D}$ is the set of transactions in the dataset.*
2. *$R = \{T_{\mathcal{D}}(W) \mid W \subseteq \mathcal{I}\}$ is a family of sets of transactions such that for each itemset $W \subseteq \mathcal{I}$, the set $T_{\mathcal{D}}(W) = \{\tau \in \mathcal{D} \mid W \subseteq \tau\}$ of all transactions containing W is an element of R.*

Theorem 2. *Let $\mathcal{D}$ be a dataset and let $S = (X, R)$ be the associated range space. Let $d \in \mathbb{N}$. Then $\mathsf{VC}(S) \geq d$ if and only if there exists a set $\mathcal{A} \subseteq \mathcal{D}$ of d transactions from $\mathcal{D}$ such that for each subset $\mathcal{B} \subseteq \mathcal{A}$ of $\mathcal{A}$, there exists an itemset $I_{\mathcal{B}}$ such that 1) all transactions in $\mathcal{B}$ contain $I_{\mathcal{B}}$ and 2) no transaction $\rho \in \mathcal{A} \setminus \mathcal{B}$ contains $I_{\mathcal{B}}$.*

Corollary 1. *Let $\mathcal{D}$ be a dataset and $S = (\mathcal{D}, R)$ be the corresponding range space. Then, the VC-Dimension $\mathsf{VC}(S)$ of S, is the maximum integer d such that there is a set $\mathcal{A} \subseteq \mathcal{D}$ of d transactions from $\mathcal{D}$ such that for each subset $\mathcal{B} \subseteq \mathcal{A}$ of $\mathcal{A}$, there exists an itemset $I_{\mathcal{B}}$ such that 1) all transactions in $\mathcal{B}$ contain $I_{\mathcal{B}}$ and 2) no transaction $\rho \in \mathcal{A} \setminus \mathcal{B}$ contains $I_{\mathcal{B}}$.*

Computing the exact VC-dimension of a dataset is extremely expensive from a computational point of view. This does not come as a suprise, as it is known that computing the VC-dimension of a range space (X, R) can take time $O(|R||X|^{\log |R|})$ [23, Thm. 4.1]. It is instead possible to give an upper bound to the VC-dimension of a dataset, and a procedure to efficiently compute the bound.

Definition 12. *Let $\mathcal{D}$ be a dataset. The* d-index *of a dataset is defined as the maximum integer d such that $\mathcal{D}$ contains at least d transactions of length at least d.*

A note of folklore: if the dataset represents the scientific publications of a given scientist, with transactions corresponding to articles and items in a transaction corresponding to the citations received by the paper, then the d-index of the dataset corresponds to the h-index of the scientist.

The d-index d of a dataset $\mathcal{D}$ can be computed in one scan of the dataset and with total memory $O(d)$. The scan starts with $d^* = 1$ and it stores the length of the first transaction. At any given step the procedure stores d^*, the current estimate of d, computed as the maximum d' such that the the scan up to this step found at least d' transactions with length at least d', and keeps a list of the sizes of the transactions longer than d' found so far. There can be no more than d' such transactions. As the scan proceeds, the procedure updates d^* and the list of transactions sizes greater than d^*.

The d-index is an upper bound to the VC-dimension of a dataset.

Theorem 3. *Let $\mathcal{D}$ be a dataset with d-index d. Then the range space $S = (X, R)$ corresponding to $\mathcal{D}$ has VC-dimension at most d.*

This bound is strict, i.e., there are indeed datasets with VC-dimension exactly d, as formalized by the following Theorem.

Theorem 4. *There exists a dataset $\mathcal{D}$ with d-index d and such the corresponding range space has VC-dimension exactly d.*

The datasets built in the proof of Thm. 4 are extremely artificial. Our experiments suggest that the VC-dimension of real datasets is usually much smaller than the upper bound presented in Thm. 3.

4 Mining (top-K) Frequent Itemsets and Association Rules

We apply the VC-dimension results to constructing efficient sampling algorithms with performance guarantees for approximating the collections of FI's, top-K FI's and AR's.

4.1 Mining Frequent Itemsets

We construct bounds for the sample size needed to obtain relative/absolute ε-close approximations to the collection of FI's. The algorithms to compute the approximations use a standard exact FI's mining algorithm on the sample, with an appropriately adjusted minimum frequency threshold, as formalized in the following lemma.

Lemma 1. *Let $\mathcal{D}$ be a dataset with transactions built on a ground set $\mathcal{I}$, and let d be the d-index of $\mathcal{D}$. Let $0 < \varepsilon, \delta < 1$. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ with size $|\mathcal{S}| = \min\{|\mathcal{D}|, \frac{4c}{\varepsilon^2}\left(d + \log\frac{1}{\delta}\right)\}$, for some constant c. Then $\mathsf{FI}(\mathcal{S}, \mathcal{I}, \theta - \frac{\varepsilon}{2})$ is an absolute ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ with probability at least $1 - \delta$.*

One very interesting consequence of this result is that we do not need to know the minimum frequency threshold θ in advance to build the sample: the properties of the ε-approximation allow to use the same sample for any threshold and for different thresholds, i.e., the sample does not need to be rebuilt if we want to mine it with a threshold θ first and with another threshold θ' later.

It is important to note that the VC-dimension of a dataset, and therefore the sample size from (2) needed to probabilistically obtain an ε-approximation, is independent from the size (number of transactions) in $\mathcal{D}$ and also of the size of $\mathsf{FI}(\mathcal{S}, \mathcal{I}, \theta)$. It only depends on the quantity d, which is always less or equal to the length of the longest transaction in the dataset, which in turn is less or equal to the number of different items $|\mathcal{I}|$.

To obtain a relative ε-close approximation, we need to add a dependency on θ as shown in the following Lemma.

Lemma 2. *Let $\mathcal{D}$, d, ε, and δ as in Lemma 1. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ with size $|\mathcal{S}| = \min\{|\mathcal{D}|, \frac{4(2+\varepsilon)c}{\varepsilon^2\theta(2-\varepsilon)}\left(d\log\frac{2+\varepsilon}{\theta(2-\varepsilon)} + \log\frac{1}{\delta}\right)\}$, for some constant c. Then $\mathsf{FI}(\mathcal{S}, \mathcal{I}, (1 - \frac{\varepsilon}{2})\theta)$ is a relative ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ with probability at least $1 - \delta$.*

4.2 Mining Top-K Frequent Itemsets

Given the equivalence $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K) = \mathsf{FI}(\mathcal{D},\mathcal{I},f_{\mathcal{D}}^{(K)})$, we could use the above FI's sampling algorithms if we had a good approximation of $f_{\mathcal{D}}^{(K)}$, the threshold frequency of the top-K FI's.

For the absolute ε-close approximation we first execute a standard top-K FI's mining algorithm on the sample to estimate $f_{\mathcal{D}}^{(K)}$ and then run a standard FI's mining algorithm on the same sample using a minimum frequency threshold depending on our estimate of $f_{\mathcal{S}}^{(K)}$. Lemma 3 formalizes this intuition.

Lemma 3. *Let $\mathcal{D}$, d, ε, and δ be as in Lemma 1. Let K be a positive integer. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ with size $|\mathcal{S}| = \min\{|\mathcal{D}|, \frac{16c}{\varepsilon^2}\left(d + \log\frac{1}{\delta}\right)\}$, for some constant c, then $\mathsf{FI}(\mathcal{S},\mathcal{I},f_{\mathcal{S}}^{(K)} - \frac{\varepsilon}{2})$ is an absolute ε-close approximation to $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$ with probability at least $1-\delta$.*

Note that as in the case of the sample size required for an absolute ε-close approximation to $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$, we do not need to know K in advance to compute the sample size for obtaining an absolute ε-close approximation to $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$.

Two different samples are needed for computing a relative ε-close approximation to $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$, the first one to compute a lower bound to $f_{\mathcal{D}}^{(K)}$, the second to extract the approximation. Details for this case are presented in Lemma 4.

Lemma 4. *Let $\mathcal{D}$, d, ε, and δ be as in Lemma 1. Let K be a positive integer. Let δ_1,δ_2 be two reals such that $(1-\delta_1)(1-\delta_2) \geq (1-\delta)$. Let $\mathcal{S}_1$ be a random sample of $\mathcal{D}$ with some size $|\mathcal{S}_1| = \frac{\phi c}{\varepsilon^2}\left(d + \log\frac{1}{\delta_1}\right)$ for some $\phi > 2\sqrt{2}/\varepsilon$ and some constant c. If $f_{\mathcal{S}_1}^{(K)} \geq \frac{2\sqrt{2}}{\varepsilon\phi}$, then let $p = \frac{2-\varepsilon}{2+\varepsilon}\theta$ and let $\mathcal{S}_2$ be a random sample of $\mathcal{D}$ of size $\min\{|\mathcal{D}|, \frac{4c}{\varepsilon^2 p}(d\log\frac{1}{p} + \log\frac{1}{\delta})\}$ for some constant c. Then $\mathsf{FI}(\mathcal{S}_2,\mathcal{I},(1-\varepsilon/2)(f_{\mathcal{S}_1}^{(K)} - \varepsilon/\sqrt{2\phi}))$ is a relative ε-close approximation to $\mathsf{TOPK}(\mathcal{D},\mathcal{I},K)$ with probability at least $1-\delta$.*

4.3 Mining Association Rules

Our final theoretical contribution concerns the discovery of relative/absolute approximations to $\mathsf{AR}(\mathcal{D},\mathcal{I},\theta,\eta)$ from a sample. Lemma 5 builds on a result from [6, Sect. 5] and covers the *relative* case, while Lemma 6 deals with the *absolute* one.

Lemma 5. *Let $0 < \delta,\varepsilon,\theta,\gamma < 1$, $\phi = \max\{3, 2-\varepsilon+2\sqrt{1-\varepsilon}\}$, $\eta = \frac{\varepsilon}{\phi}$, and $p = \frac{1-\eta}{1+\eta}\theta$. Let $\mathcal{D}$ be a dataset with d-index d. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ of size $\min\{|\mathcal{D}|, \frac{c}{\eta^2 p}(d\log\frac{1}{p} + \log\frac{1}{\delta})\}$ for some constant c. Then $\mathsf{AR}(\mathcal{S},\mathcal{I},(1-\eta)\theta,\frac{1-\eta}{1+\eta}\gamma)$ is a relative ε-close approximation to $\mathsf{AR}(\mathcal{D},\mathcal{I},\theta,\gamma)$ with probability at least $1-\delta$.*

Lemma 6. *Let $\mathcal{D}$, d, θ, γ, ε, and δ be as in Lemma 5 and let $\varepsilon_{\mathrm{rel}} = \frac{\varepsilon}{\max\{\theta,\gamma\}}$.*

Fix $\phi = \max\{3, 2-\varepsilon_{\mathrm{rel}}+2\sqrt{1-\varepsilon_{\mathrm{rel}}}\}$, $\eta = \frac{\varepsilon_{\mathrm{rel}}}{\phi}$, and $p = \frac{1-\eta}{1+\eta}\theta$. Let $\mathcal{S}$ be a random sample of $\mathcal{D}$ of size $\min\{|\mathcal{D}|, \frac{c}{\eta^2 p}(d\log\frac{1}{p} + \log\frac{1}{\delta})\}$ for some constant c. Then $\mathsf{AR}(\mathcal{S},\mathcal{I},(1-\eta)\theta,\frac{1-\eta}{1+\eta}\gamma)$ is an absolute ε-close approximation to $\mathsf{AR}(\mathcal{D},\mathcal{I},\theta,\gamma)$.

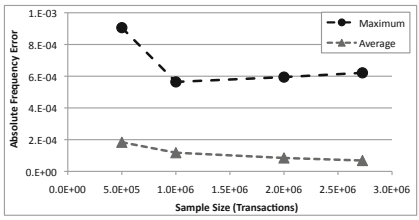
Note that the sample size needed for absolute ε-close approximations to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ depends on θ and γ, which was not the case for absolute ε-close approximations to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ and $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$.

5 Experimental Evaluation

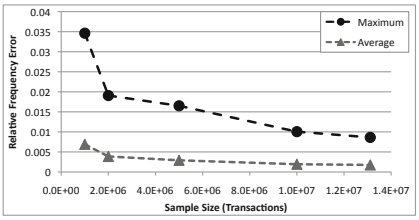
In this section we present an extensive experimental evaluation of our methods to extract approximations of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$, and $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$. Due to space constraints, we focus on a subset of the results.

Our first goal is to evaluate the *quality* of the approximations obtained using our methods, by comparing the experimental results to the analytical bounds. We also evaluate how strict the bounds are by testing whether the same quality of results can be achieved at sample sizes smaller than those suggested by the theoretical analysis. We then show that our methods can significantly speed-up the mining process, fulfilling the motivating promises of the use of sampling in the market basket analysis tasks. Lastly, we compare the sample sizes from our results to the best previous work [6].

We tested our methods on both real and artificial datasets. The real datasets come from the FIMI'04 repository (`http://fimi.ua.ac.be/data/`). Since most of them have a moderately small size, we replicated their transactions a number of times, with the only effect of increasing the size of the dataset but no change in the distribution of the frequencies of the itemsets. The artificial datasets were built such that their corresponding range spaces have VC-dimension equal to the maximum transaction length d, which is the maximum possible as shown in Thm. 3. To create these datasets, we followed the proof of Thm. 4 and used the generator included in ARtool (`http://www.cs.umb.edu/~laur/ARtool/`), which is similar to the one presented in [2]. We used the the FP-Growth and Apriori implementations in ARtool to extract frequent itemsets and association rules. In all our experiments we fixed $\delta = 0.1$. In the experiments involving absolute (resp. relative) ε-close approximations we set $\varepsilon = 0.01$ (resp. $\varepsilon = 0.05$). The constant c was fixed to 0.5 as suggested by [24]. For each dataset we selected a range of minimum frequency thresholds and a set of values for K when extracting the top-K frequent itemsets. For association rules discovery we set the minimum confidence threshold $\gamma \in \{0.5, 0.75, 0.9\}$. For each dataset and each combination of parameters we created random samples with size as suggested by our theorems and with smaller sizes to evaluate the strictness of the bounds. We measured, for each set of parameters, the *absolute frequency error* and the *absolute confidence error*, defined as the error $|f_{\mathcal{D}}(X) - f_{\mathcal{S}}(X)|$ (resp. $|c_{\mathcal{D}}(Y) - c_{\mathcal{S}}(Y)|$) for an itemset X (resp. an association rule Y) in the approximate collection extracted from sample $\mathcal{S}$. When dealing with the problem of extracting *relative* ε-close approximations, we defined the *relative frequency error* to be the absolute frequency error divided by the real frequency of the itemset and analogously for the relative confidence error (dividing by the real confidence). In the figures we plot the maximum and the average for these quantities, taken over all itemsets or association rules in the output collection. In order to limit the influence of a single sample, we computed and plot in the figures the maximum (resp. the average) of these quantities in three runs of our methods on three different samples for each size.



(a) Absolute Frequency Error



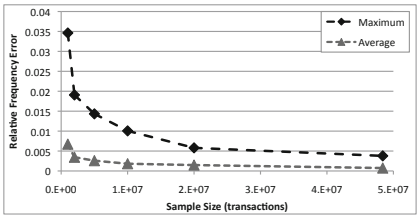
(b) Relative Frequency Error

Fig. 1. Absolute / Relative ε-close Approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$

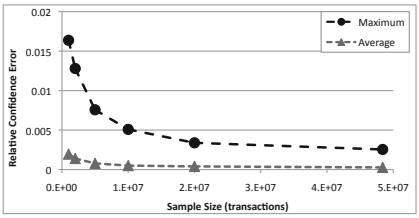
The first important result of our experiments is that, for all problems, for every combination of parameters and every run, the collection of itemsets or of association rules obtained using our methods always satisfied the requirements to be an absolute or relative ε-close approximation to the real collection. Thus in practice our methods indeed achieve or exceed the theoretical guarantees for approximations of the collections $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, \theta)$, and $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$.

Evaluating the strictness of the bounds to the sample size was the second goal of our experiments. In Figure 1a we show the behaviour of the maximum frequency error as function of the sample size in the itemsets obtained from samples using the method presented in Lemma 1 (i.e., we are looking for an *absolute* ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$). The rightmost plotted point corresponds to the sample size suggested by the theoretical analysis. We are showing the results for the dataset BMS-POS replicated 40 times (d-index $d = 134$), mined with $\theta = 0.02$. It is clear from the picture that the guaranteed error bounds are achieved even at sample sizes smaller than what suggested by the analysis and that the error at the sample size derived from the theory (rightmost plotted point for each line) is one to two orders of magnitude smaller than the maximum tolerable error $\varepsilon = 0.01$. This fact seems to suggest that there is still room for improvement in the bounds to the sample size needed to achieve an absolute ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$. In Fig. 1b we report similar results for the problem of computing a *relative* ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ for an artificial dataset whose range space has VC-dimension d equal to the length of the longest transaction in the dataset, in this case 33. The dataset contained 100 million transactions. The sample size, suggested by Lemma 2, was computed using $\theta = 0.01, \varepsilon = 0.05$, and $\delta = 0.1$. The conclusions we can draw from the results for the behaviour of the relative frequency error are similar to those we got for the absolute case. For the case of absolute and relative ε-close approximation to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, K)$, we observed similar results, which we do not report here because of space constraints.

The results of the experiments to evaluate our method to extract a relative ε-close approximation to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$ are presented in Fig. 2a and 2b. The same observations as before hold for the relative frequency error, while it is interesting to note that the relative confidence error is even smaller than the frequency error, most possibly because the confidence of an association rule is the ratio between the frequencies of two itemsets that appear in the same transactions and their sample frequencies will therefore have similar errors that cancel out when the ratio is computed. Similar conclusions can be made for the absolute ε-close case (not reported due to space constraints).



(a) Relative Frequency Error



(b) Relative Confidence Error

Fig. 2. Relative ε-close approximation to $\mathsf{AR}(\mathcal{D}, \mathcal{I}, \theta, \gamma)$

The major motivating intuition for the use of sampling in market basket analysis tasks is that mining a sample of the dataset is faster than mining the entire dataset. Nevertheless, the mining time does not only depend on the number of transactions, but also on the number of frequent itemsets. Given that our methods suggest to mine the sample at a lowered minimum frequency threshold, this may cause an increase in running time that would make our method not useful in practice, because there may be many more frequent itemsets than at the original frequency threshold. We performed a number of experiments to evaluate whether this was the case and present the results in Fig. 3. We mined the artificial dataset introduced before for different values of θ, and created samples of size sufficient to obtain a relative ε-close approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, for $\varepsilon = 0.05$ and $\delta = 0.1$. Figure 3 shows the time needed to mine the large dataset and the time needed to create and mine the samples. It is possible to appreciate that, even considering the sampling time, the speed up achieved by our method is relevant, proving the usefulness of sampling.

Comparing our results to previous work we note that the bounds generated by our technique are always linear in the VC-dimension d associated with the dataset. As reported in Table 1, the best previous work [6] presented bounds that are linear in the maximum transaction size Δ for two of the six problems studied here. Figures 4a and 4b shows a comparison of the actual sample sizes for relative ε-close approximations to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ for as function of θ and ε. To compute the points for these figures, we set $\Delta = d = 50$, corresponding to the worst possible case for our method, i.e., when the VC-dimension of the range space associated to the dataset is exactly equal to the maximum transaction length. We also fixed $\delta = 0.05$ (the two methods behave equally as δ changes). For Fig. 4a, we fixed $\varepsilon = 0.05$, while for Fig. 4b we fixed $\theta = 0.05$. From the

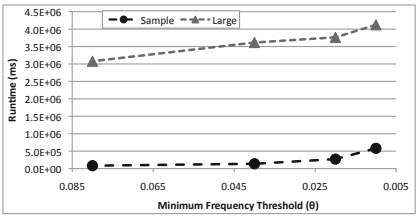
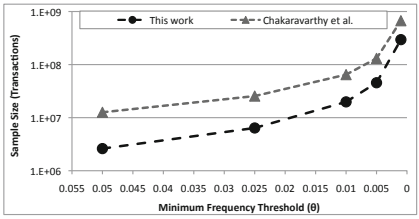
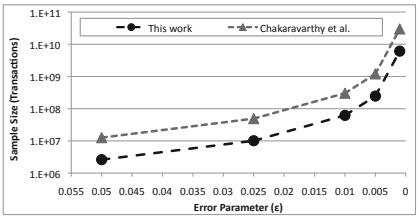


Fig. 3. Runtime Comparison. The sample line includes the sampling time.



(a) Sample size as function of θ



(b) Sample size as function of ε

Fig. 4. Sample sizes for relative ε-close approximations to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$

Table 2. Values for maximum transaction length Δ and d-index d for real datasets

	accidents	BMS-POS	BMS-Webview-1	kosarak	mushroom	pumsb*	retail	webdocs
Δ	51	164	267	2497	23	63	76	71472
d	46	81	57	443	23	59	58	2452

Figures we can appreciate that both bounds have similar, but not equal, dependencies on θ and ε. More precisely the bound presented in this work is less dependent on ε and only slightly more dependent on θ. It also evident that the sample sizes suggested by the bound presented in this work are always much smaller than those presented in [6] (the vertical axis has logarithmic scale). In this comparison we used $\Delta = d$, but almost all real datasets we encountered have $d \ll \Delta$ as shown in Table 2 which would result in a larger gap between the sample sizes provided by the two methods.

References

[1] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. SIGMOD Rec. 22, 207–216 (1993)
[2] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB 1994 (1994)
[3] Alon, N., Spencer, J.H.: The Probabilistic Method, 3rd edn. Wiley (2008)
[4] Brönnimann, H., Chen, B., Dash, M., Haas, P., Scheuermann, P.: Efficient data reduction with ease. In: KDD 2003 (2003)
[5] Ceglar, A., Roddick, J.F.: Association mining. ACM Comput. Surv. 38(5) (2006)
[6] Chakaravarthy, V.T., Pandit, V., Sabharwal, Y.: Analysis of sampling techniques for association rule mining. In: ICDT 2009 (2009)
[7] Chandra, B., Bhaskar, S.: A new approach for generating efficient sample from market basket data. Expert Sys. with Appl. 38(3), 1321–1325 (2011)
[8] Chazelle, B.: The discrepancy method: randomness and complexity, Cambridge (2000)
[9] Chen, B., Haas, P., Scheuermann, P.: A new two-phase sampling based algorithm for discovering association rules. In: KDD 2002 (2002)
[10] Chen, C., Horng, S.-J., Huang, C.-P.: Locality sensitive hashing for sampling-based algorithms in association rule mining. Expert Sys. with Appl. 38(10), 12388–12397 (2011)
[11] Cheung, Y.-L., Fu, A.W.-C.: Mining frequent itemsets without support threshold: With and without item constraints. IEEE Trans. on Knowl. and Data Eng. 16, 1052–1069 (2004)

[12] Chuang, K.-T., Chen, M.-S., Yang, W.-C.: Progressive Sampling for Association Rules Based on Sampling Error Estimation. In: Adv. in Knowl. Disc. and Data Mining. Springer, Heidelberg (2005)
[13] Chuang, K.-T., Huang, J.-L., Chen, M.-S.: Power-law relationship and self-similarity in the itemset support distribution: analysis and applications. The VLDB Journal 17(5) (2008)
[14] Fu, A.W.-C., Kwong, R.W.-W., Tang, J.: Mining N-most Interesting Itemsets. In: Ohsuga, S., Raś, Z.W. (eds.) ISMIS 2000. LNCS (LNAI), vol. 1932, pp. 59–67. Springer, Heidelberg (2000)
[15] Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Min. Knowl. Discov. 15, 55–86 (2007)
[16] Har-Peled, S., Sharir, M.: Relative (p, ε)-approximations in geometry. Discr. & Comput. Geometry 45(3), 462–496 (2011)
[17] Hu, X., Yu, H.: The Research of Sampling for Mining Frequent Itemsets. In: Wang, G.-Y., Peters, J.F., Skowron, A., Yao, Y. (eds.) RSKT 2006. LNCS (LNAI), vol. 4062, pp. 496–501. Springer, Heidelberg (2006)
[18] Hwang, W., Kim, D.: Improved association rule mining by modified trimming. In: CIT 2006 (2006)
[19] Jia, C., Lu, R.: Sampling Ensembles for Frequent Patterns. In: Wang, L., Jin, Y. (eds.) FSKD 2005. LNCS (LNAI), vol. 3613, pp. 1197–1206. Springer, Heidelberg (2005)
[20] Jia, C.-Y., Gao, X.-P.: Multi-scaling sampling: An adaptive sampling method for discovering approximate association rules. J. of Comp. Sci. and Tech. 20, 309–318 (2005)
[21] John, G.H., Langley, P.: Static versus dynamic sampling for data mining. In: KDD 1996 (1996)
[22] Li, Y., Gopalan, R.: Effective Sampling for Mining Association Rules. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 391–401. Springer, Heidelberg (2004)
[23] Linial, N., Mansour, Y., Rivest, R.L.: Results on learnability and the Vapnik-Chervonenkis dimension. Information and Computation 1, 33–49 (1991)
[24] Löffler, M., Phillips, J.M.: Shape Fitting on Point Sets with Probability Distributions. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 313–324. Springer, Heidelberg (2009)
[25] Mahafzah, B.A., Al-Badarneh, A.F., Zakaria, M.Z.: A new sampling technique for association rule mining. J. of Information Science 3, 358–376 (2009)
[26] Mampaey, M., Tatti, N., Vreeken, J.: Tell me what i need to know: succinctly summarizing data with itemsets. In: KDD 2011 (2011)
[27] Mannila, H., Toivonen, H., Verkamo, I.: Efficient algorithms for discovering association rules. In: KDD 1994 (1994)
[28] Parthasarathy, S.: Efficient progressive sampling for association rules. In: ICDM 2002 (2002)
[29] Pietracaprina, A., Riondato, M., Upfal, E., Vandin, F.: Mining top-K frequent itemsets through progressive sampling. Data Min. Knowl. Discov. 21, 310–326 (2010)
[30] Pietracaprina, A., Vandin, F.: Efficient Incremental Mining of Top-K Frequent Closed Itemsets. In: Corruble, V., Takeda, M., Suzuki, E. (eds.) DS 2007. LNCS (LNAI), vol. 4755, pp. 275–280. Springer, Heidelberg (2007)
[31] Riondato, M., Upfal, E.: Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. CoRR abs/1111.6937 (2011)
[32] Scheffer, T., Wrobel, S.: Finding the most interesting patterns in a database quickly by using sequential sampling. J. Mach. Learn. Res. 3, 833–862 (2002)
[33] Toivonen, H.: Sampling large databases for association rules. In: VLDB 1996 (1996)
[34] Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag (1999)
[35] Vapnik, V.N., Chervonenkis, A.J.: On the uniform convergence of relative frequencies of events to their probabilities. Theory of Prob. and its Appl. 16(2), 264–280 (1971)

[36] Vasudevan, D., Vojonović, M.: Ranking through random sampling. MSR-TR-2009-8 8, Microsoft Research (2009)
[37] Wang, J., Han, J., Lu, Y., Tzvetkov, P.: TFP: An efficient algorithm for mining top-k frequent closed itemsets. IEEE Trans. on Knowl. and Data Eng. 17, 652–664 (2005)
[38] Wang, S., Dash, M., Chia, L.-T.: Efficient Sampling: Application to Image Data. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 452–463. Springer, Heidelberg (2005)
[39] Zaki, M., Parthasarathy, S., Li, W., Ogihara, M.: Evaluation of sampling for data mining of association rules. In: RIDE 1997 (1997)
[40] Zhang, C., Zhang, S., Webb, G.I.: Identifying approximate itemsets of interest in large databases. Applied Intelligence 18, 91–104 (2003)
[41] Zhao, Y., Zhang, C., Zhang, S.: Efficient frequent itemsets mining by sampling. In: AMT 2006 (2006)

Smoothing Categorical Data

Arno Siebes and René Kersten

Universiteit Utrecht, The Netherlands
arno@cs.uu.nl, renegaa@hotmail.com

Abstract. Global models of a dataset reflect not only the large scale structure of the data distribution, they also reflect small(er) scale structure. Hence, if one wants to see the large scale structure, one should somehow subtract this smaller scale structure from the model.

While for some kinds of model – such as boosted classifiers – it is easy to see the "important" components, for many kind of models this is far harder, if at all possible. In such cases one might try an implicit approach: simplify the data distribution without changing the large scale structure. That is, one might first smooth the local structure out of the dataset. Then induce a new model from this smoothed dataset. This new model should now reflect the large scale structure of the original dataset. In this paper we propose such a smoothing for categorical data and for one particular type of models, viz., code tables.

By experiments we show that our approach preserves the large scale structure of a dataset well. That is, the smoothed dataset is simpler while the original and smoothed datasets share the same large scale structure.

1 Introduction

Most often data has structure across multiple scales. It is relatively easy to see fine-grained structure, e.g., through pattern mining. The lower the "support" of a pattern the more detailed structure it conveys. Since the large scale structure of data is convoluted with small scale structure it is, unfortunately, less easy to see its large scale structure. Simply focussing on "high support" patterns might miss large scale structure; we will show examples of this later.

Global models of the data, on the other hand, attempt to capture all, relevant, aspects of the data distribution. This often encompasses both global structure as well as local structure. If the type of model used is additive, such as a boosted classifier [5], the large scale structure may be approximated by disregarding small weight components. For non-additive types of models, syntactic operations that reveal the large scale structure are far less obvious.

This is unfortunate, since this large scale structure conveys important insight both in the data distribution and in the model. Hence, the problem we research in this paper: how to get insight in the large scale structure of a dataset?

If manipulating the model is difficult, manipulating the data might be easier. That is, *smoothing* the local structure – as described by the model – from the data while retaining the global structure – as described by the model. Inducing a new

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 42–57, 2012.

model from the smoothed dataset should then reveal the large scale structure of the data distribution as described by the original model. This is our approach.

Note that we should consistently write "structure of the data as described by the model", as we did above, but we simplify this to "structure of the data".

Smoothing is a well-known statistical technique [14] mostly for numerical data and to a lesser extend for ordered data. In this paper we focus on *categorical* data, for which as far as we are aware, no previous smoothing methods exist. Succinctly, our goal is to smooth out local structure from standard categorical data tables, while preserving large scale structure.

The structure of a categorical dataset is given by *item sets* [1]; in the context of categorical data, an item is defined as an attribute-value pair. The large scale structure is mostly – but not exclusively, as noted above – given by more frequent item sets. The small scale structure is given by less frequent item sets.

Informally, two equally sized datasets are indistinguishable if the support of all item sets is the same on both datasets. In the same vein, two equally sized datasets are similar if the support of most item sets is more or less the same on the two datasets. Again informally, we say that two equally sized datasets have the same large scale structure if they are similar as far as *frequent* item sets are concerned. This is made precise in Section 3.

To smooth the data while retaining the large scale structure, we need to use a model. For it is this model that describes the large scale structure we want to preserve. Hence, the way to smooth depends on the type of model. In this paper we focus on one particular class of models that consists of itemsets, viz. *code tables* as introduced in [11].

A code table consists of item sets associated with codes and can be used to encode a database; see Section 2. What is important here is that a code table implicitly encodes a probability distribution on all possible tuples in the database. By replacing less likely tuples with more likely tuples, the small scale structure is smoothed from the dataset while the large scale structure is maintained. That is, the support of most frequent item sets is not changed too much by these replacements; see Section 3.

Note that the reason why we don't simply use the set of all frequent item sets to do the smoothing is not only that this set is far too large. It also doesn't imply a straight forward distribution on all possible tuples; if only because of the interdependencies between the supports of different item sets.

In Figure 1 the effect of this smoothing is illustrated on the *mammals* dataset. This dataset consists of presence/absence records of European mammals[1] within geographical areas of 50x50 kilometres. In the left map, we depict the occurrence of the "item set" {European Hare, European Polecat, European Mole, Wild Boar}, with dark blue dots. The other dots denote the occurrence of variants of this set; variants that differ in one mammal. No dot means that neither the item set, nor any of these variants occur in that place.

[1] The full version of the mammal dataset is available for research purposes upon request from the Societas Europaea Mammalogica. http://www.european-mammals.org

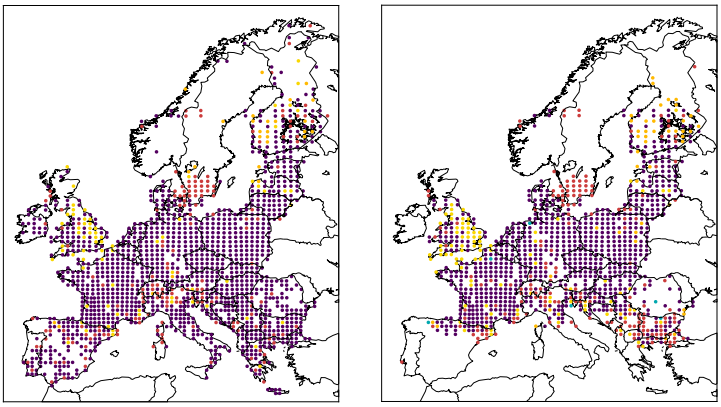


Fig. 1. The smoothed mammals dataset depicted on the right is far more homogeneous than the original mammals dataset depicted on the left

The right map shows the effect of smoothing. The map looks far more homogeneous: the Balkans changed from a variety of colours to almost uniformly dark blue. This does not happen in England and Sweden, but not because the map of the latter two looks more homogeneous than the map of the Balkans. Rather, this happens because the total set of mammals that occur in the Balkans resemble the set of mammals that occur in the rest of mainland Europe far more than the sets of mammals that occur in England and south Sweden do.

2 Preliminaries

2.1 Data and Patterns

We assume that the dataset is a standard rectangular table, well known from relational databases.That is we have a finite set of attributes $\mathcal{A} = \{A_1, \ldots, A_m\}$. Moreover, we assume that each attribute A_i has a finite, categorical, domain Dom_i. A tuple t over $\mathcal{A}$ is an element of $Dom = \prod_{i=1}^{m} Dom_i$. A database D over $\mathcal{A}$ is a bag of tuples over $\mathcal{A}$.

Since our work is rooted in item set mining [1], we will also use the terminology from that area. This means firstly that we will talk about transactions rather than tuples. Secondly we have a set of items $\mathcal{I} = \{I_1, \ldots I_n\}$. Each item I_j corresponds to an attribute-value pair (A_i, v_i^k), where $v_i^k \in Dom_i$. Note that this implies that all transactions have the same number of items.

A transaction t supports an item $I = (A_i, v_i^k)$ iff $t.A_i = v_i^k$. As usual, the support of an item set $J \subset \mathcal{I}$ in D, denoted by $sup_D(J)$, is defined as the number of transactions in D that support all items in J. Given a user defined threshold for support, denoted by *min-sup* or θ, an item set J is called *frequent* on D iff $sup_D(J) \geq \theta$. All frequent item sets can be found relatively efficiently [1].

2.2 Introducing KRIMP

The key idea of the KRIMP algorithm is the code table. A code table is a two-column table that has item sets on the left-hand side and a code for each item set on its right-hand side. The item sets in the code table are ordered descending on 1) item set length, 2) support size and 3) lexicographically. The actual codes on the right-hand side are of no importance but their lengths are. To explain how these lengths are computed, the coding algorithm needs to be introduced.

A transaction t is encoded by KRIMP by searching for the first item set I in the code table for which $I \subseteq t$. The code for I becomes part of the encoding of t. If $t \setminus I \neq \emptyset$, the algorithm continues to encode $t \setminus I$. Since it is insisted that each code table contains at least all singleton item sets, this algorithm gives a unique encoding to each (possible) transaction over $\mathcal{I}$. The set of item sets used to encode a transaction is called its *cover*.

The length of the code of an item in a code table CT depends on the database we want to compress; the more often a code is used, the shorter it should be. To compute this code length, we encode each transaction in the database D. The *usage* of an item set $I \in CT$, denoted by $usage(I)$ is the number of transactions $t \in D$ which have I in their cover. That is, $usage(I) = |\{t \in D \mid I \in cover(t)\}|$.

For an $I \in CT$, the probability that I is used to encode an arbitrary $t \in D$, is simply the fraction of its usage, i.e.,

$$P(I \mid D) = \frac{usage(I)}{\sum_{J \in CT} usage(J)}$$

For optimal compression of D, the higher $P(I)$, the shorter its code should be. Given that a prefix code is necessary for unambiguous decoding, the well-known optimal Shannon code [4] is used. We now have the length of an item set I encoded with CT defined as $L(I \mid CT) = -\log(P(I \mid D))$.

The length of the encoding of a transaction is now simply the sum of the code lengths of the item sets in its cover:

$$L(t \mid CT) = \sum_{I \in cover(t,CT)} L(I \mid CT)$$

The size of the encoded database is the sum of the sizes of the encoded transactions:

$$L(D \mid CT) = \sum_{t \in D} L(t \mid CT) = -\sum_{I \in CT} usage(I) \log\left(\frac{usage(I)}{\sum_{J \in CT} usage(J)}\right)$$

To find the best code table for a dataset, the Minimum Description Length (MDL) principle [6] is used. Which can be roughly described as follows.

Given a set of models $\mathcal{H}$, the best model $H \in \mathcal{H}$ is the one that minimises $L(H) + L(D \mid H)$, in which $L(H)$ is the length, in bits, of the description of H, and $L(D \mid H)$ is the length, in bits, of D encoded with H. One can paraphrase this by: the smaller $L(H) + L(D|H)$, the better H models D.

To apply MDL to code tables, we still need to define the size of a code table, as we previously did in [11]. We only count those item sets that have a non-zero usage. The size of the right-hand side column is obvious; it is the sum of all the different code lengths. For the size of the left-hand side column, note that the simplest valid code table consists only of the singleton item sets. This is the *standard encoding (ST)*, which we use to compute the size of the item sets in the left-hand side column. Hence, the size of code table CT is given by:

$$L(CT \mid D) = \sum_{I \in CT : usage(I) \neq 0} L(I \mid ST) + L(I \mid CT)$$

An optimal code table is a code table which minimises:

$$L(D, CT) = L(CT \mid D) + L(D \mid CT)$$

Finally, $\mathcal{L}(D) = L(D, CT)$ for an optimal code table CT for D.

Unfortunately, computing an optimal code table is intractable [11], hence we introduced the heuristic algorithm Krimp. Krimp starts with a valid code table (only the collection of singletons) and a sorted list of candidates (frequent item sets). These candidates are assumed to be sorted descending on 1) support size, 2) item set length and 3) lexicographically. Each candidate item set is considered by inserting it at the right position in CT and calculating the new total compressed size. A candidate is only kept in the code table iff the resulting total size is smaller than it was before adding the candidate. If it is kept, all other elements of CT are reconsidered to see if they still positively contribute to compression; see [11].

3 The Problem

As stated in the Introduction, our goal is to make the large scale structure of a data distribution more visible by smoothing out small(er) scale structure. The formalisation of that goal rests on four considerations.

Consideration 1: A code table models the structure in the data by a very small subset of all (frequent) item sets, chosen because together they compress the database well. The usages of these item sets say something about their importance with respect to large scale structure, but not everything. High usage clearly means large scale structure, but low usage does *not* necessarily mean small scale structure. The reason for this is the order of the code table which is used to compute covers. In other words, to "see" the large scale structure of the dataset, it is not enough to simply focus on item sets with a high usage. We'll return to this observation later in this paper when we discuss our experiments.

Consideration 2: We claim that the structure of a categorical dataset is given by the support of all item sets; independent from the type of model used. The motivation for this claim is based on two observations.

The first is that if two equally sized datasets D_1 and D_2 over $\mathcal{I}$ have the same support for all item sets over $\mathcal{I}$, they are row permutations of each other. That is,

there exists a permutation π of the rows such that $\pi(D_1) = D_2$. This is obvious from the fact that the transactions in D_1 and D_2 are item sets themselves.

The second observation is that D_1 and D_2 will be indistinguishable by any type of statistical analysis [2]. For, all such analysis boils down to computing aggregates computed on subtables of a dataset. Given that these subtables correspond to item sets, equal support means equal results of the analysis.

Consideration 3: Statistical computations are, in general, robust. That is, small changes in the input yield small changes in the output. One reason for this is that two samples from the same distribution will invariably be subtly different; to be useful, statistical analysis should "smooth out" such differences.

In other words, if D_1 and D_2 have almost the same support for all item sets, statistical analysis will mostly imply that D_1 and D_2 are indistinguishable. For our purposes we can be even more tolerant, for we are willing – indeed aiming – to lose some (local) structure. That is, we are satisfied if for most item sets the support is almost the same. We can loosely formalise this by requiring that the support of a random item set is almost the same on both data sets.

Consideration 4: There are two weak points in this formulation. Firstly, that it is a statement about all item sets. Surely, if it is large scale structure we are interested in, item sets with (very) low support are unimportant. Secondly, what is almost the same?

Fortunately, these two weak points can be resolved in one step. The fact that we are not interested in low support item sets simply means that we are interested in *frequent* item sets. That is, item sets whose support is at least θ. If structure that is described by item sets with a support smaller than θ is deemed not interesting, we should also not worry too much about differences in support smaller than θ.

3.1 Formalising the Problem

Given all the previous, we have the following definition.

Definition 1. *Let D_1 and D_2 be two datasets over $\mathcal{I}$ and let $\epsilon, \delta \in \mathbb{R}_{\geq 0}$. D_2 is (ϵ, δ)-similar to D_1 if for a random item set I with $sup_{D_1}(I) \geq \delta$*

$$P(|sup_{D_1}(I) - sup_{D_2}(I)| \geq \delta) \leq \epsilon$$

If D_2 is to be a smoothed version of D_1, then we want it to be (ϵ, δ)-similar to D_1 for some $\epsilon, \delta \in \mathbb{R}_{\geq 0}$. But we also want D_2 to be simpler than D_1, i.e., we want D_2 to have a simpler code table than D_1.

In our MDL approach, it is easy to formalise what it means that D_1 is simpler than D_2. If D_1 and D_2 have the same size (the same number of transactions and, thus, the same number of items) and $\mathcal{L}(D_2) < \mathcal{L}(D_1)$, then D_2 is simpler than D_1. The reason is that, in this case, D_2 has less local structure that D_1. Local structure next to global structure makes a dataset harder to compress. Hence, we have the following definition.

Definition 2. *Let D_1 and D_2 be two equal sized (categorical) datasets over $\mathcal{I}$. D_2 is simpler than D_1 iff*

$$\mathcal{L}(D_2) < \mathcal{L}(D_1)$$

That a dataset D_2 which is both simpler than D_1 in this sense and (ϵ, δ)-similar to D_1, also has a simpler code table than D_1 is something only experiments can show.

Data Smoothing Problem
Let D be a dataset over $\mathcal{I}$ and let $\epsilon, \delta \in \mathbb{R}_{\geq 0}$. Moreover, let $\mathcal{D}_D^{(\epsilon,\delta)}$ be the set of all datasets over $\mathcal{I}$ that have the same number of transactions as D and are (ϵ, δ)-similar to D. Find a $D' \in \mathcal{D}_D^{(\epsilon,\delta)}$ that minimises $\mathcal{L}(D')$.

4 Introducing Smooth

Given that both the set of datasets with the same number of transactions as D and the set of code tables are finite, our problem is clearly decidable. However, given that finding an optimal code table for a dataset is already intractable [11], our current problem is also intractable. Hence we have to resort to heuristics.

For this heuristic we use an observation first made in [8]: a code table CT on a database D implicitly defines a probability distribution on the set of all possible tuples in the domain Dom of D. Let t be such an arbitrary tuple, then we can compress it with CT:

$$\begin{aligned} L(t \mid CT) &= \sum_{I \in cover(t)} L(I \mid CT) = \sum_{I \in cover(t)} -\log(P(I \mid D)) \\ &= -\log\left(\prod_{I \in cover(t)} P(I \mid D)\right) = -\log(P(t \mid D)) \stackrel{def}{=} -\log(P(t \mid CT)) \end{aligned}$$

The one but last equation rests on the Naive Bayes like assumption that the item sets in a cover are independent. They are not(!), but in previous work this distribution has shown to characterise the data distribution on D very well [8,13]. Hence, we decided to use it here as well.

The main idea of the Smooth algorithm is to replace less likely tuples in D with more likely tuples, both according to this probability distribution. This strategy is based on the following observation: if D is changed into D' by replacing one $t \in D$ by a $t' \in Dom$ such that $L(t' \mid CT) < L(t \mid CT)$, then $L(D', CT) < L(D, CT)$. That is D' is simpler than D, according to CT.

However, if we replace an arbitrary $t \in D$ by an equally arbitrary $t' \in Dom$ which compresses better, there is no guarantee that D' will be (ϵ, δ)-similar to D for the given parameters ϵ and δ. Drastic changes could influence the support of many (frequent) item sets drastically, effectively disturbing the data distribution captured by the code table, not only on small scales, but also on large scales. To maintain the large scale balance we take two precautions:

Algorithm 1. SMOOTH$(D, CT, \epsilon, \delta)$

$D' := D$
$F :=$ frequent item sets, min-sup is δ
while D' is (ϵ, δ)-similar to D for F **do**
  choose $t \in D'$ according to $P_{sel}(t \mid CT)$
  choose $t' \in Var(t)$ according to $P_{var}(t' \mid CT)$
  $D' := (D' \setminus \{t\}) \cup \{t'\}$
end while
return D'

- In one step we only consider modifications with edit distance one. That is, only one attribute-value is changed in one tuple. This set of variants of t is denoted by $Var(t)$.
- Both the selection of tuples to modify and their modification is random, where the choice is guided by the code table.
  - For the tuple selection, the probability of selecting $t \in D$, denoted by $P_{sel}(t \mid CT)$, is defined by

$$P_{sel}(t \mid CT) = \frac{(P(t \mid CT))^{-1}}{\sum_{t' \in D}(P(t \mid CT))^{-1}}$$

  - The probability of selecting an alternative $t' \in Var(t)$, denoted by $P_{var}(t')$, is defined by

$$P_{var}(t' \mid CT) = \frac{P(t' \mid CT)}{\sum_{t'' \in Var(t)} P(t'' \mid CT)}$$

We choose the tuple to replace at random using P_{sel} for two reasons. First, because in this way we "disturb" the data distribution as little as possible. Second, if we only attempt to replace tuples with a large encoded size, we more easily get stuck at a local optimum. We use P_{var} to select a variant at random, again because this disturbs the original data distribution as little as possible.

Neither of these two precautions guarantees that D' will be (ϵ, δ)-similar to D. They only heighten the probability that *one* replacement will result in a (ϵ, δ)-similar dataset. If we would let this replacement scheme run long enough, the resulting database would in most cases *not* be (ϵ, δ)-similar to D. For example, if $t \in D$ is unique in having the shortest encoded length, the replacement scheme, if left to run unbounded, would converge on a dataset that only contains copies of t. Therefore SMOOTH also checks whether D' is still (ϵ, δ)-similar to D.

SMOOTH is listed in algorithm 1. Apart from a dataset D and parameters ϵ and δ, it also takes a code table CT as input.

5 Experiments

In this section we report on two sets of experiments. The first set of experiments, on some well-known UCI datasets[2] (transformed using [3]), shows that SMOOTH

[2] http://archive.ics.uci.edu/ml/

achieves its goal. That is, it results in simpler datasets and, more importantly, simpler models that are still good models of the original dataset.

While these experiments shed some light on how SMOOTH achieves these goals, the second set of experiments is especially designed to do that. We take an artificial dataset which we corrupt by noise, and by examining how SMOOTH alters the noisy dataset we gain a deeper understanding of its doing.

In all experiments, we use KRIMP to approximate the optimal code table for dataset D and use that as input for SMOOTH.

5.1 UCI Data

For all experiments on the UCI datasets Iris, Pageblocks, Pima, Wine, Led7, and TicTacToe we used $\epsilon = 0.01$ and a minimal support, and thus δ, of 5. The experiments were also performed with $\delta = 1$ and $\delta = 10$, but given that the results are very similar we do not report on them here. For all experiments - except for the classification experiments - we report the averages of 50 repeats.

Table 1. Compressed sizes for $\epsilon = 0.01$, averaged over 50 repeats; standard deviation $< 2\%$

Dataset	L(D, CT)	L(D', CT')	L(D', CT)
Iris	1685	1500	1572
Pageblocks	11404	10883	11031
Pima	9331	8652	8864
Wine	11038	10090	9980
Led7	30867	30664	30085
TicTacToe	29004	28575	27956

The results in Table 1 show that D' is indeed simpler than D. In all cases $L(D', CT')$ is significantly smaller than $L(D, CT)$, Moreover, CT is still a good code table for D', which can be seen from the fact that $L(D', CT') < L(D', CT) < L(D, CT)$.

Table 2 shows how many changes SMOOTH made to the original dataset before $\hat{\epsilon}$ (the measured value for ϵ) exceeded 0.01 for the first time. Also, it shows the value of $\hat{\epsilon}$ at that time. Only few changes lead to the simpler datasets shown in Table 1.

The effects of these changes is shown in Table 3. In that table we compare the number of non-singleton patterns in the code tables, the length of these patterns, and the percentage of the datasets covered by these patterns. In all cases, the number of non-singleton item sets in CT' is smaller than the number of of non-singleton item sets in CT. At the same time, the average size of these patterns goes up. That is, the code table has become simpler. The database has also become simpler, as the number of unique rows goes down. The joint effect of these two simplifications can be seen in the last two columns: the percentage of items in

Table 2. The number of changes made and $\hat{\epsilon}$ when it first exceeded 0.01, averaged over 50 repeats

Dataset	# changes	$\hat{\epsilon}$
Iris	9.7 ± 1.1	0.014 ± 0.002
Pageblocks	81.2 ± 13.7	0.016 ± 0.005
Pima	34.8 ± 5.0	0.014 ± 0.003
Wine	162.9 ± 17.7	0.011 ± 0.001
Led7	52.8 ± 7.0	0.012 ± 0.001
TicTacToe	279.0 ± 23.8	0.011 ± 0.000

Table 3. Comparing the number of non-singleton patterns in the code tables, the length of these patterns, the number of unique rows in the data sets, and the percentage of the datasets covered by these patterns; averaged over 50 repeats

	# of patterns		avg pattern length		unique rows		% of covered items	
Dataset	CT	CT'	CT	CT'	D	D'	CT on D	CT' on D'
Iris	14	13.3 ± 0.6	3.4	3.7 ± 0.1	42	35.6 ± 0.9	91	95
Pageblocks	43	32.1 ± 1.4	8.3	10.3 ± 0.2	72	50.5 ± 2.4	100	100
Pima	56	48.9 ± 2.2	4.9	5.7 ± 0.2	170	157.7 ± 2.1	96	97
Wine	60	53.8 ± 3.0	3.5	3.8 ± 0.2	177	176.9 ± 0.2	67	72
Led7	153	148.3 ± 5.8	6.6	6.9 ± 0.1	326	298.8 ± 4.2	98	99
TicTacToe	159	155.0 ± 6.7	4.0	4.1 ± 0.1	958	905.2 ± 6.3	90	91

the database that is covered by non-singleton item sets from the code table goes up, although not by very much.

Both the compression results in Table 1 and the measured ϵ value for $\delta = 5$ in Table 2 indicate that D' has a data distribution very similar to D. For an independent verification of this claim we also performed classification experiments with these code tables. The basic set-up is the same as in [8], with 25-fold cross-validation. Each class-database was individually smoothed with SMOOTH ($\epsilon = 0.01$); the test data was, of course, *not smoothed*. The results are in Table 4.

The first observation is that for PageBlocks, Pima, Led7, and TicTacToe, the classification results of the simplified code table are on par with those of the original code table. Moreover, for all but Pima, these results are way above the baseline scores (assign the tuple to the largest class). The somewhat disappointing result on Pima is caused by the fact that it has small classes that are too small to learn well using MDL.

Table 4. Classification accuracy on independent original data, 25-fold cross-validation.

	Classification accuracy		
Dataset	Baseline	CT	CT'
Iris	33.3	94.7 ± 9.3	90.0 ± 15.2
Pageblocks	69.8	92.5 ± 1.5	92.3 ± 1.4
Pima	65.1	69.5 ± 6.9	69.5 ± 9.1
Wine	39.3	91.6 ± 12.0	79.9 ± 21.0
Led7	11.0	73.8 ± 3.5	74.1 ± 3.2
TicTacToe	65.3	87.8 ± 7.3	80.4 ± 8.4

The second observation is that the degradation in classification performance is far larger for Iris and Wine. Again this is caused by dataset size, both Iris and Wine are small datasets. Moreover, even the degraded scores are way above baseline.

Note that these results by no means imply that SMOOTH induces a state-of-the-art classifier. Rather, the results reconfirm that SMOOTH yields characteristic code tables of the *original* dataset, i.e., structure is preserved.

A classification scheme based on code tables might be biased towards SMOOTH. After all, one of the design goals of SMOOTH was not to change code tables too much. To verify that the original data distribution is not changed too much, we also performed these classification experiments with some well-known algorithms as implemented in Weka [7]. The set-up is the same as with the SMOOTH

Table 5. Classification accuracy before and after smoothing. In all cases, $\epsilon = 0.01$, $\delta \in \{1, 5, 10\}$, and 'orig' denotes the original (non-smoothed) dataset

Dataset - δ	C4.5	Ripper	LR	NB	SVM
Iris-orig	92.67 ± 9.66	92.67 ± 9.66	90.00 ± 13.05	94.00 ± 7.98	92.00 ± 10.80
Iris-1	92.67 ± 9.66	94.00 ± 9.66	92.00 ± 9.84	94.67 ± 6.89	92.00 ± 10.80
Iris-5	98.00 ± 6.32	98.00 ± 6.32	96.00 ± 6.44	94.67 ± 6.89	94.67 ± 7.57
Iris-10	95.33 ± 7.06	95.33 ± 7.06	94.67 ± 6.13	94.67 ± 6.13	94.67 ± 6.13
Led7-orig	75.19 ± 2.90	72.09 ± 3.82	75.63 ± 3.31	75.59 ± 2.81	75.91 ± 2.94
Led7-1	75.16 ± 2.83	71.88 ± 3.14	75.63 ± 3.32	75.50 ± 2.90	75.91 ± 2.96
Led7-5	75.28 ± 4.07	72.31 ± 3.56	75.59 ± 3.92	75.47 ± 3.83	75.75 ± 4.01
Led7-10	74.94 ± 3.66	72.78 ± 4.06	74.78 ± 4.11	75.28 ± 4.05	75.38 ± 4.11
Pageblocks-orig	92.64 ± 1.42	92.57 ± 1.36	92.75 ± 1.49	92.68 ± 1.45	91.91 ± 1.67
Pageblocks-1	92.66 ± 1.40	92.51 ± 1.32	92.75 ± 1.43	92.66 ± 1.47	91.91 ± 1.68
Pageblocks-5	92.64 ± 2.14	92.51 ± 2.17	92.69 ± 2.14	92.60 ± 2.14	91.80 ± 2.30
Pageblocks-10	92.68 ± 1.58	92.57 ± 1.64	92.60 ± 1.65	92.66 ± 1.66	91.82 ± 1.60
Pima-orig	74.58 ± 3.76	73.29 ± 5.14	72.52 ± 4.85	74.32 ± 4.70	73.55 ± 5.82
Pima-1	74.58 ± 3.76	73.81 ± 4.69	72.39 ± 4.93	74.19 ± 4.66	73.68 ± 5.79
Pima-5	74.32 ± 6.00	74.06 ± 6.42	72.77 ± 6.97	74.32 ± 6.68	73.68 ± 6.22
Pima-10	74.84 ± 7.57	74.06 ± 6.81	73.55 ± 8.43	74.58 ± 7.17	73.16 ± 7.32
TicTacToe-orig	85.21 ± 3.92	97.92 ± 1.96	98.02 ± 2.11	70.21 ± 4.23	87.71 ± 4.39
TicTacToe-1	85.00 ± 6.04	97.92 ± 1.70	98.12 ± 1.61	70.21 ± 4.00	87.08 ± 3.78
TicTacToe-5	84.17 ± 3.64	98.33 ± 1.22	97.50 ± 1.32	70.94 ± 3.31	84.90 ± 4.34
TicTacToe-10	81.15 ± 3.91	97.08 ± 2.29	96.88 ± 1.90	71.04 ± 3.50	82.92 ± 3.71
Wine-orig	90.56 ± 6.44	88.89 ± 6.93	92.78 ± 6.95	95.00 ± 4.86	87.78 ± 5.74
Wine-1	85.56 ± 9.15	87.22 ± 5.89	93.89 ± 8.47	95.00 ± 4.86	91.11 ± 5.97
Wine-5	86.11 ± 5.40	88.89 ± 11.11	91.11 ± 7.94	93.89 ± 4.10	88.33 ± 7.15
Wine-10	82.22 ± 9.37	84.44 ± 8.61	91.67 ± 6.00	93.89 ± 8.05	87.78 ± 10.08

based classification, that is we did 25-fold cross-validation, running SMOOTH on each train set, while, again, *the test set was not smoothed*. Table 5 shows the results for C4.5, Ripper, Logistic Regression, Naive Bayes, and Support Vector Machines, each with their default settings in Weka.

The important observation is that the accuracy doesn't change significantly when the dataset is smoothed. Whether δ is set to 1, 5, or 10, the difference in accuracy is not significantly different from the baseline; where the baseline is the accuracy of the algorithm on the original (non-smoothed) dataset. This is true for the rule-based classifiers C4.5 and Ripper, for the instance-based classifier NB, for the traditional statistical classifier Logistic Regression, and for the SVM classifier. None of these classifiers detects a significant difference between the original data distribution and the smoothed data distribution.

There is one notable exception to this observation: C4.5 on Wine. There we see a notable, significant, drop in accuracy. While inspecting the resulting trees, we noticed serious overfitting. Using the default settings only takes you so far.

A natural question with these results is: do the other classifiers also become simpler on the smoothed datasets? While this was not a design goal for SMOOTH - after all, it is a *model-driven* approach - we inspected this for the Ripper results.

The average number of rules goes down while the average length of the rules goes up slightly. This is very similar to the changes we encountered earlier for the code table elements, which also became fewer and longer.

5.2 Artificial Data

The results of the previous subsection already show some of SMOOTH's effects, but even more insight can be gained when we know the ideal result beforehand.

For this, we created an artificial dataset as follows. We generated 50 unique tuples randomly (over 7 attributes, each with a domain from 4 to 18 values). Each of these 50 tuples was duplicated between 10 and 30 times randomly, such that the end result was a dataset with 1000 tuples. This is the Clean dataset. We then ran KRIMP on Clean (with min-sup=1), and the result was - unsurprisingly - a code table with 50 item sets, one for each unique tuple in Clean.

For the experiments, we randomly replace items in the dataset with others (of course, while still adhering to the attributes' domains). The items to be replaced are chosen uniformly. We vary the amount of error to investigate the smoothing capability of SMOOTH in comparison to the amount of noise in the data.

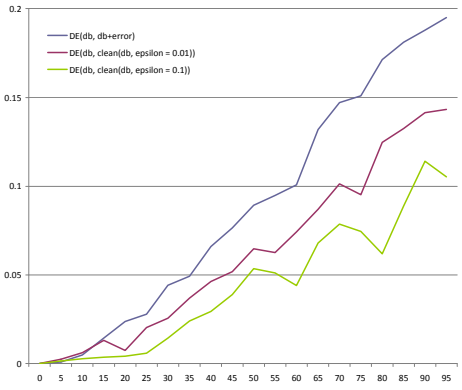


Fig. 2. The x-axis depicts the number of errors made on Clean, the y-axis gives $\hat{\epsilon}$, i.e, the probability of an error larger than δ

Note that we do *not* perform this experiment to show that SMOOTH is good at removing noise. After all, data points that are considered noise in one setting may be considered perfectly valid data in another setting. However, in our artificial setting noise and local structure happen to coincide and we know exactly what the noise is. Hence, we here use SMOOTH as a noise remover, because it allows us to investigate how it removes local structure.

With this proviso, SMOOTH is run to try to "clean up" the data. The parameters are: $\delta = 1$, and $\epsilon = 0.01$ and 0.1 respectively. The algorithm is automatically stopped after 1000 iterations, because for low amounts of error added, epsilon can never even attain 0.01; for larger amounts of noise it can, of course. Performance is measured by comparing the noisy and smoothed datasets to the original Clean dataset in three different ways.

The first measure is simply the measured epsilon, i.e. $\hat{\epsilon}$, for $\delta = 1$. This to make sure we can see the 'noise' being added and subsequently removed. The obtained measurements are graphed in Figure 2. Results are depicted for three more or less noisy datasets, the first being the noisy dataset itself. Obviously, this has consistently the highest $\hat{\epsilon}$. The other two datasets are smoothed versions

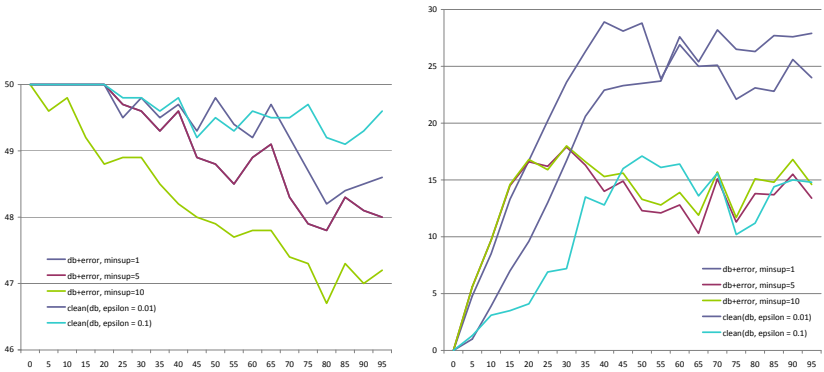


Fig. 3. In the figure on the left $|CT' \cap CT|$ is graphed for various datasets. In the figure on the right $|CT' - CT|$ is graphed for the same datasets. In both, the x-axis depicts the number of errors made on Clean.

of the noisy dataset, with ϵ set to 0.01 resp. 0.1. Note that the ϵ parameter is set with regard to the *noisy* dataset and thus determines how far SMOOTH can go away from the noisy dataset. The two graphs show that SMOOTH uses this extra wriggle-room to get closer to the Clean dataset: the graph for $\epsilon = 0.1$ is consistently better than the graph for $\epsilon = 0.01$.

Note, however, that the algorithm is not able to drop $\hat{\epsilon}$ on the cleaned dataset to 0, even if $\hat{\epsilon}$ between the clean and the cluttered dataset is less than the ϵ parameter. This is because there is always a chance that the algorithm will select and adjust a row with no noise on it. However, the difference between the low chance of adjusting a clean row in a bad way and the high chance of adjusting a noisy row in a good way ensures that $\hat{\epsilon}$ will still drop significantly; in Figure 2 the error is halved!

In Figure 3 the other two measures are graphed, both versus the number of errors on Clean across different datasets. The left figure depicts the number of the original patterns left (50 is ideal). This is the part of the original structure that can still be seen in CT' and we denote this by $|CT' \cap CT|$. The right figure depicts the number of wrongfully added patterns in CT'. These patterns are present in CT' but not in CT; they clutter up the code table and therefore obscure the true structure in the data. We denote this measure by $|CT' - CT|$.

Firstly note that adding more noise to the data will cause more of the good patterns to be 'broken up' into smaller patterns. Therefore $|CT' \cap CT|$ drops and $|CT' - CT|$ rises. Secondly note that increasing the minimal support, i.e, δ, will filter those smaller patterns out. These patterns are simply not frequent enough and thus they cannot end-up in the code table. But for this reason, we also cannot recover the patterns in the Clean dataset that were lost, as can be seen in the left figure. As in Figure 2, $\epsilon = 0.1$ outperforms $\epsilon = 0.01$ for both measures and for the same reason: SMOOTH needs some leeway to remove the unwanted, local structure.

6 Discussion

The reason to introduce the SMOOTH algorithm was to smooth the local structure from the data while retaining the global structure, such that a simpler, but still characteristic, code table could be mined from this smoothed dataset. The results from the previous section show that that goal has been reached.

Firstly, both the code table and the database get simpler when SMOOTH is applied. This is, e.g., clear from Table 3. The new code table has fewer, larger, item sets and the dataset has fewer unique tuples. Moreover, the new code table describes the structure of the new database better for a larger number of items is covered by non-singleton item sets. This latter fact is also witnessed by Table 1, as CT' compresses D' better than CT compresses D.

Secondly, D' - and thus CT' - retains important structure from D. This is, again, witnessed by Table 1: CT compresses D' better than D. Moreover, Table 2 shows that SMOOTH achieves these goals for a modest epsilon, implying that D and D' are almost indistinguishable. Further witness that D and CT' retain the important structure in D and CT is given by Table 4 and Table 5. Both show that training on D and D' leads to classifiers statistically indistinguishable on an independent test from D, for a wide variety of classification algorithms! Moreover, Table 4 shows that CT' retains the important structure in D, for CT' is almost as good as CT in classifying tuples from the original distribution D.

Finally, SMOOTH achieves these results by removing local structure. Figure 2 shows this very well. For a dataset with 10% noise, SMOOTH with $\epsilon = 0.1$ (measured against the noisy dataset) achieves $\hat{\epsilon} = 0.1$ with regard to the Clean dataset. The number of errors with regard to the Clean dataset is halved!

An even stronger witness for the claim that large scale structure is retained while local structure is removed is given by Figure 3. These two graphs show firstly that SMOOTH not only retains global structure, it even recovers structure that is present in the original dataset, but not visible in the corrupted dataset. Secondly, these graphs show that local structure is indeed removed, as the number of wrongfully added patterns goes down.

So, SMOOTH achieves its results very well. The reader might, however, wonder if these results couldn't be achieved easier. One easier way is to simply mine with a larger minimal support, after all the large scale structure is mostly given by (very) frequent item sets. Unfortunately, this doesn't work. Figure 3 shows that a higher minimal support means that fewer original patterns are recovered; we may miss large scale structure that is obfuscated by local structure.

If concentrating on (very) frequent item sets doesn't work, a second simpler strategy might seem to concentrate on item sets with a high usage. That is, simply smooth out the low usage patterns from the code table by modifying only tuples with a large code size. In fact, this strategy suffers from the same problem as the first alternative; Figure 3 shows that original large scale structure may be obfuscated by low usage item sets.

Note that the latter observations also illustrate the importance of the problem solved in this paper: if we do not smooth the data, there may be large scale structure in the data that is simply not apparent from the code table.

7 Related Work

Data smoothing is a research area with a rich history in areas such as statistics and image processing, but also in e.g. signal processing where it is known as filtering. Giving an overview of this vast field is far outside the scope of this paper. A good introductory book from the viewpoint of Statistics is [14].

Smoothing usually refers to continues operations. If the data is real valued, smoothing is often performed by convoluting the data with some distribution. Even if the data is discrete (but still numerical), convolution is often the weapon of choice. For ordered categorical data, a Poisson regression model with log-likelihood may be used [12]. We are not aware of any papers that address general categorical data and/or take an approach similar to ours. The big difference is that convolution – and regression – always involve all data, where there may be many transactions in a dataset that SMOOTH doesn't even touch.

Within the field of pattern mining, our approach is related to fault tolerant patterns [9]. Roughly speaking, these are patterns which with some small modifications to the data would get a larger support. We do discover such patterns, that is what the modifications that SMOOTH makes to the database imply. There is, however, a large difference in aim. Fault tolerant pattern miners want to find all fault tolerant patterns. We want to discover the global structure of the data and discover some fault tolerant patterns as a by product.

In our own research, our paper on missing data [13] is related to this paper. In there we design three algorithms that complete a database with missing data. Like SMOOTH, these algorithms use a probability distribution on variants. There is one major difference, though. With missing data, one knows exactly where the problem is. Thus, the imputation algorithms introduced in [13] do not have to select "which tuples to modify where" and can run until convergence. In the current case, unfortunately, we do not know where the problem is. Only by slowly massaging the data do we discover its, sometimes hidden, large scale structure.

Related in goal, but not algorithmically is our introduction of a structure function in [10]. In that paper we introduce a series of models that capture ever finer details of the data distribution. The first model looks at a very coarse scale, while the final model looks at a very fine scale. The major difference with SMOOTH is that in [10] the dataset is not changed. This means that the structure function approach will not uncover structure which is hidden by noise. That is, while SMOOTH is – to a certain level – able to reconstruct the original dataset from a corrupted variant, the structure function of [10] is not able to do that.

8 Conclusions

The observation that triggered the research reported on in this paper is that it is often hard to understand the large scale structure of the data from a model. The approach we take is that we smooth the local structure from the dataset – guided by the model – while retaining the large scale structure. A model induced from this smoothed dataset reflects the large scale structure of the original data.

While smoothing is a well known for numerical and ordered data, this paper introduces a smoothing algorithm, called SMOOTH, for categorical data.

SMOOTH uses code tables such as, e.g., generated by KRIMP to smooth the data. It smoothes the data by gradually modifying tuples in the database. While smoothing it ensures that the large scale structure is maintained, i.e., that the support of most frequent item sets remains the same. At the same time it ensures that the data set becomes simpler, i.e., that it compresses better.

The experiments show that SMOOTH works well. Both the smoothed data set and its code table are simpler than the originals. Moreover, both datasets give more or less the same support to most item sets. Hence, both datasets have more or less the same structure. This is further corroborated by the fact that the original dataset and the smoothed dataset lead to equally good classifiers for an independent test set of the original(!) dataset. This observation was shown to hold for a large variety of classification algorithms.

Finally, experiments on artificial data, for which we know the ideal outcome, show that SMOOTH does what it is supposed to do. The large scale structure is retained while local structure is removed. Even if the large scale structure is hidden by local structure it may be recovered by SMOOTH.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI (1996)
2. Agresti, A.: Categorical Data Analysis, 2nd edn. Wiley (2002)
3. Coenen, F.: The LUCS-KDD discretised/normalised (C)ARM data library (2003)
4. Cover, T., Thomas, J.: Elements of Information Theory, 2nd edn. Wiley (2006)
5. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
6. Grünwald, P.D.: Minimum description length tutorial. In: Grünwald, P., Myung, I. (eds.) Advances in Minimum Description Length. MIT Press (2005)
7. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: Weka data mining software: An update. SIGKDD Explorations 11 (2009)
8. 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)
9. Pei, J., Tung, A.K.H., Han, J.: Fault tolerant pattern mining: Problems and challenges. In: DMKD (2001)
10. Siebes, A., Kersten, R.: A structure function for transaction data. In: Proc. SIAM Conf. on Data Mining (2011)
11. Siebes, A., Vreeken, J., van Leeuwen, M.: Item sets that compress. In: Proc. SIAM Conf. Data Mining, pp. 393–404 (2006)
12. Simonoff, J.S.: Three sides of smoothing: Categorical data smoothing, nonparametric regression, and density estimation. International Statistical Reviews /Revue Internationale de Statistique 66(2), 137–156 (1998)
13. Vreeken, J., Siebes, A.: Filling in the blanks - krimp minimization for missing data. In: Proceedings of the IEEE International Conference on Data Mining (2008)
14. Wand, M., Jones, M.: Kernel Smoothing. Chapman & Hall (1994)

An Experimental Comparison of Hybrid Algorithms for Bayesian Network Structure Learning

Maxime Gasse, Alex Aussem, and Haytham Elghazel

Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France

Abstract. We present a novel hybrid algorithm for Bayesian network structure learning, called Hybrid HPC (H2PC). It first reconstructs the skeleton of a Bayesian network and then performs a Bayesian-scoring greedy hill-climbing search to orient the edges. It is based on a subroutine called HPC, that combines ideas from incremental and divide-and-conquer constraint-based methods to learn the parents and children of a target variable. We conduct an experimental comparison of H2PC against Max-Min Hill-Climbing (MMHC), which is currently the most powerful state-of-the-art algorithm for Bayesian network structure learning, on several benchmarks with various data sizes. Our extensive experiments show that H2PC outperforms MMHC both in terms of goodness of fit to new data and in terms of the quality of the network structure itself, which is closer to the true dependence structure of the data. The source code (in *R*) of H2PC as well as all data sets used for the empirical tests are publicly available.

1 Introduction

A Bayesian network (BN) is a probabilistic model formed by a structure and parameters. The structure of a BN is a directed acyclic graph (DAG), whilst its parameters are conditional probability distributions associated with the variables in the model. The graph of a BN itself is an independence map, which is very useful for many applications, including feature selection [2,18,24] and inferring causal relationships from observational data [11,2,4,5,8,6]. The problem of finding the DAG that encodes the conditional independencies present in the data attracted a great deal of interest over the last years [23,26,27,15,22,36,21].

Ideally the DAG should coincide with the dependence structure of the global distribution, or it should at least identify a distribution as close as possible to the correct one in the probability space. This step, called structure learning, is similar in approaches and terminology to model selection procedures for classical statistical models. Basically, constraint-based (CB) learning methods systematically check the data for conditional independence relationships and use them as constraints to construct a partially oriented graph representative of a BN equivalence class, whilst search-and-score (SS) methods make use of a goodness-of-fit

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 58–73, 2012.

score function for evaluating graphical structures with regard to the data set. Hybrid methods attempt to get the best of both worlds: they learn a skeleton with a CB approach and constrain on the DAGs considered during the SS phase. There are many excellent treatments of BNs which survey the learning methods (see [16] for instance).

Both CB and SS approaches have advantages and disadvantages. CB approaches are relatively quick, deterministic, and have a well defined stopping criterion; however, they rely on an arbitrary significance level to test for independence, and they can be unstable in the sense that an error early on in the search can have a cascading effect that causes many errors to be present in the final graph. SS approaches have the advantage of being able to flexibly incorporate users' background knowledge in the form of prior probabilities over the structures and are also capable of dealing with incomplete records in the database (e.g. EM technique). Although SS methods are favored in practice when dealing with small dimensional data sets, they are slow to converge and the computational complexity often prevents us from finding optimal BN structures [22]. With currently available exact algorithms [14,29] and a decomposable score like BDeu, the computational complexity remains exponential, and therefore, such algorithms are intractable for BNs with more than around 30 vertices on current workstations [15]. For larger sets of variables, the computational burden becomes prohibitive. With this in mind, the ability to restrict the search locally around the target variable is a key advantage of CB methods over SS methods. They are able to construct a local graph around the target node without having to construct the whole BN first, hence their scalability [18,24,23,35,20].

With a view to balancing the computation cost with the desired accuracy of the estimates, several hybrid methods have been proposed recently. Tsamardinos et al. [35] proposed the Min-Max Hill Climbing (MMHC) algorithm and conducted one of the most extensive empirical comparison performed in recent years showing that MMHC was the fastest and the most accurate method in terms of structural error based on the structural hamming distance. More specifically, MMHC outperformed both in terms of time efficiency and quality of reconstruction the PC [30], the Sparse Candidate [12], the Three Phase Dependency Analysis [9], the Optimal Reinsertion [17], the Greedy Equivalence Search [10], and the Greedy Hill-Climbing Search on a variety of networks, sample sizes, and parameter values. Although MMHC is rather heuristic by nature (it returns a local optimum of the score function), MMHC is currently considered as the most powerful state-of-the-art algorithm for BN structure learning capable of dealing with thousands of nodes in reasonable time. With a view to enhance its performance on small dimensional data sets, Perrier et al. [22] proposed recently a hybrid algorithm that can learn an *optimal* BN (i.e., it converges to the true model in the sample limit) when an undirected graph is given as a structural constraint. They defined this undirected graph as a super-structure (i.e., every DAG considered in the SS phase is compelled to be a subgraph of the super-structure). This algorithm can learn optimal BNs containing up to 50 vertices when the average degree of the super-structure is around two, that is, a sparse

structural constraint is assumed. To extend its feasibility to BN with a few hundred of vertices and an average degree up to four, [15] proposed to divide the super-structure into several clusters and perform an optimal search on each of them in order to scale up to larger networks. Despite interesting improvements in terms of score and structural hamming distance on several benchmark BNs, they report running times about 10^3 times longer than MMHC on average, which is still prohibitive in our view.

Therefore, there is great deal of interest in hybrid methods capable of improving the structural accuracy of both CB and SS methods on graphs containing up to thousands of vertices. However, they make the strong assumption that the skeleton (also called super-structure) contains at least the edges of the true network and as small as possible extra edges. While controlling the false discovery rate (i.e., false extra edges) in BN learning has attracted some attention recently [3,20,34], to our knowledge, there is no work on controlling actively the rate of false-negative errors (i.e., false missing edges).

In this study, we compare MMHC with a another hybrid algorithm for BN structure learning, called Hybrid HPC (H2PC). H2PC and MMHC share exactly the same SS procedure to allow for fair comparisons; the only difference lies in the procedure for learning the skeleton (i.e., the undirected graph given as a structural constraint to the SS search). While MMHC is based on Max-Min Parents and Children (MMPC) to learn the parents and children of a variable, H2PC is based on a subroutine called Hybrid Parents and Children (HPC), that combines ideas from incremental and divide-and-conquer CB methods. The ability of HPC and MMPC [35] to infer the parents and children set of candidate nodes was assessed in [23] through several empirical experiments. In this work, we conduct an experimental comparison of H2PC against Max-Min Hill-Climbing (MMHC) on several benchmarks and various data sizes.

2 Preliminaries

Formally, a BN is a tuple $< \mathbb{G}, P >$, where $\mathbb{G} =< \mathbf{U}, \mathbf{E} >$ is a directed acyclic graph (DAG) whose nodes represent the variables in the domain $\mathbf{U}$, and whose edges represent direct probabilistic dependencies between them. P denotes the joint probability distribution on $\mathbf{U}$. The BN structure encodes a set of conditional independence assumptions: that each node X_i is conditionally independent of all of its non descendants in $\mathbb{G}$ given its parents $\mathbf{Pa}_i^{\mathbb{G}}$. These independence assumptions, in turn, imply many other conditional independence statements, which can be extracted from the network using a simple graphical criterion called d-separation [19].

We denote by $X \perp_P Y|\mathbf{Z}$ the conditional independence between X and Y given the set of variables $\mathbf{Z}$ where P is the underlying probability distribution. Note that an exhaustive search of $\mathbf{Z}$ such that $X \perp_P Y|\mathbf{Z}$ is a combinatorial problem and can be intractable for high dimension data sets. We use $X \perp_{\mathbb{G}} Y|\mathbf{Z}$ to denote the assertion that X is d-separated from Y given $\mathbf{Z}$ in $\mathbb{G}$. We denote by $\mathbf{dSep}(X, Y)$, a set that d-separates X from Y. If $< \mathbb{G}, P >$ is a BN, $X \perp_P Y|\mathbf{Z}$

if $X \perp_{\mathbb{G}} Y|\mathbf{Z}$. The converse does not necessarily hold. We say that $< \mathbb{G}, P >$ satisfies the *faithfulness condition* if the d-separations in $\mathbb{G}$ identify *all and only* the conditional independencies in P, i.e., $X \perp_P Y|\mathbf{Z}$ if and only if $X \perp_{\mathbb{G}} Y|\mathbf{Z}$. We denote by $\mathbf{PC}_X^{\mathcal{G}}$, the set of parents and children of X in $\mathcal{G}$, and by $\mathbf{SP}_X^{\mathcal{G}}$, the set of spouses of X in $\mathcal{G}$, i.e., the variables that have common children with X. These sets are unique for all $\mathcal{G}$, such that $< \mathcal{G}, P >$ satisfies the faithfulness condition and so we will drop the superscript $\mathcal{G}$.

3 Constraint-Based Structure Learning

The induction of local or global BN structures is handled by CB methods through the identification of local neighborhoods (i.e., $\mathbf{PC}_X$), hence their scalability to very high dimensional data sets. CB methods systematically check the data for conditional independence relationships in order to infer a target's neighborhood. Typically, the algorithms run either a G^2 or a χ^2 independence test when the data set is discrete and a Fisher's Z test when it is continuous in order to decide on dependence or independence, that is, upon the rejection or acceptance of the null hypothesis of conditional independence. Since we are limiting ourselves to discrete data, both the global and the local distributions are assumed to be multinomial, and the latter are represented as conditional probability tables. Conditional independence tests and network scores for discrete data are functions of these conditional probability tables through the observed frequencies $\{n_{ijk}; i = 1, \ldots, R; j = 1, \ldots, C; k = 1, \ldots, L\}$ for the random variables X and Y and all the configurations of the levels of the conditioning variables $\mathbf{Z}$. We use n_{i+k} as shorthand for the marginal $\sum_j n_{ijk}$ and similarly for n_{i+k}, n_{++k} and $n_{+++} = n$. We use a classic conditional independence test based on the mutual information. The mutual information is an information-theoretic distance measure defined as

$$MI(X, Y|\mathbf{Z}) = \sum_{i=1}^{R} \sum_{j=1}^{C} \sum_{k=1}^{L} \frac{n_{ijk}}{n} \log \frac{n_{ijk} n_{++k}}{n_{i+k} n_{+jk}}$$

It is proportional to the log-likelihood ratio test G^2 (they differ by a 2n factor, where n is the sample size). The asymptotic null distribution is χ^2 with $(R-1)(C-1)L$ degrees of freedom. For a detailed analysis of their properties we refer the reader to [1]. The main limitation of this test is the rate of convergence to its limiting distribution, which is particularly problematic when dealing with small samples and sparse contingency tables. The decision of accepting or rejecting the null hypothesis depends implicitly upon the degree of freedom which increases exponentially with the number of variables in the conditional set. Several heuristic solutions have emerged in the literature [30,23,35,33] to overcome some shortcomings of the asymptotic tests. In this study we use the two following heuristics that are used in MMHC. First, we do not perform $MI(X, Y|\mathbf{Z})$ and assume independence if there are not enough samples to achieve large enough power. We require that the average sample per count is above a user defined parameter, equal to 5, as in [35]. This heuristic is called the power rule. Second,

we consider as structural zero either case n_{+jk} or $n_{i+k} = 0$. For example, if $n_{+jk} = 0$, we consider y as a structurally forbidden value for Y when Z = z and we reduce R by 1 (as if we had one column less in the contingency table where Z = z). This is known as the degrees of freedom adjustment heuristic.

4 The Hybrid Parents and Children Algorithm (HPC)

In this section, we present a brief overview of HPC. For further details, the reader is directed to [24,23] as well as references therein. HPC (Algorithm 1) can be viewed as an ensemble method for combining many weak PC learners in an attempt to produce a stronger PC learner. HPC is based on three subroutines: *Data-Efficient Parents and Children Superset* (DE-PCS), *Data-Efficient Spouses Superset* (DE-SPS), and *Interleaved Incremental Association Parents and Children* (Inter-IAPC), a weak PC learner based on Inter-IAMB [32] that requires little computation. HPC may be thought of as a way to compensate for the large number of false negatives, at the output of the weak PC learner, by performing extra computations. It receives a target node T, a data set $\mathcal{D}$ and a set of variables $\mathbf{U}$ as input and returns an estimation of $\mathbf{PC}_T$. It is hybrid in that it combines the benefits of incremental and divide-and-conquer methods. The procedure starts by extracting a superset $\mathbf{PCS}_T$ of $\mathbf{PC}_T$ (line 1) and a superset $\mathbf{SPS}_T$ of $\mathbf{SP}_T$ (line 2) with a severe restriction on the maximum conditioning size ($\mathbf{Z} <= 2$) in order to significantly increase the reliability of the tests. A first candidate PC set is then obtained by running the weak PC learner on $\mathbf{PCS}_T \cup \mathbf{SPS}_T$ (line 3). The key idea is the decentralized search at lines 4-8 that includes, in the candidate PC set, all variables in the superset $\mathbf{PCS}_T \setminus \mathbf{PC}_T$ that have T in their vicinity. Note that, in theory, X is in the output of Inter-IAPC(Y) if and only if Y is in the output of Inter-IAPC(X). However, in practice, this may not always be true, particularly when working in high-dimensional domains. By loosening the criteria by which two nodes are said adjacent, the effective restrictions on the size of the neighborhood are now far less severe. The decentralized search has significant impact on the accuracy of HPC. It enables the algorithm to handle large neighborhoods while still being correct under faithfulness condition.

Inter-IAPC is a fast incremental method that receives a data set $\mathcal{D}$ and a target node T as its input and promptly returns a rough estimation of $\mathbf{PC}_T$, hence the term "weak" PC learner. The subroutines DE-PCS and DE-SPS (omitted for brevity) search a superset of $\mathbf{PC}_T$ and $\mathbf{SP}_T$ respectively with a severe restriction on the maximum conditioning size ($|\mathbf{Z}| <= 1$ in DE-PCS and $|\mathbf{Z}| <= 3$ in DE-SPS) in order to significantly increase the reliability of the tests. The variable filtering has two advantages : i) it allows HPC to scale to hundreds of thousands of variables by restricting the search to a subset of relevant variables, and ii) it eliminates many (almost) deterministic relationships that produce many false negative errors in the output of the algorithm. Again, the reader is encouraged to consult the papers by [24,23] for gaining more insight on these procedures.

Algorithm 1. *HPC*

Require: T: target; $\mathcal{D}$: data set; $\mathbf{U}$: the set of variables
Ensure: $\mathbf{PC}_T$: Parents and Children of T

1: $[\mathbf{PCS}_T, \mathbf{dSep}] \leftarrow$ *DE-PCS*$(T, \mathcal{D})$
2: $\mathbf{SPS}_T \leftarrow$ *DE-SPS*$(T, \mathcal{D}, \mathbf{PCS}_T, \mathbf{dSep})$
3: $\mathbf{PC}_T \leftarrow$ *Inter-IAPC*$(T, \mathcal{D}, (T \cup \mathbf{PCS}_T \cup \mathbf{SPS}_T))$
4: **for all** $X \in \mathbf{PCS}_T \setminus \mathbf{PC}_T$ **do**
5: **if** $T \in$ *Inter-IAPC*$(X, \mathcal{D}, (T \cup \mathbf{PCS}_T \cup \mathbf{SPS}_T))$ **then**
6: $\mathbf{PC}_T \leftarrow \mathbf{PC}_T \cup X$
7: **end if**
8: **end for**

Algorithm 2. *Inter-IAPC*

Require: T: target; D: data set; $\mathbf{U}$: set of variables;
Ensure: $\mathbf{PC}_T$: Parents and children of T;

1: $\mathbf{MB}_T \leftarrow \emptyset$
2: **repeat**
3: * *Add true positives to* $\mathbf{MB}_T$
4: $Y \leftarrow \mathbf{argmax}_{X \in (\mathbf{U} \setminus \mathbf{MB}_T \setminus T)} dep(T, X | \mathbf{MB}_T)$
5: **if** $T \not\perp Y | \mathbf{MB}_T$ **then**
6: $\mathbf{MB}_T \leftarrow \mathbf{MB}_T \cup Y$
7: **end if**

* *Remove false positives from* $\mathbf{MB}_T$
8: **for all** $X \in \mathbf{MB}_T$ **do**
9: **if** $T \perp X | (\mathbf{MB}_T \setminus X)$ **then**
10: $\mathbf{MB}_T \leftarrow \mathbf{MB}_T \setminus X$
11: **end if**
12: **end for**
13: **until** $\mathbf{MB}_T$ has not changed

* *Remove spouses of* T *from* $\mathbf{MB}_T$
14: $\mathbf{PC}_T \leftarrow \mathbf{MB}_T$
15: **for all** $X \in \mathbf{MB}_T$ **do**
16: **if** $\exists \mathbf{Z} \subseteq (\mathbf{MB}_T \setminus X)$ such that $T \perp X \mid \mathbf{Z}$ **then**
17: $\mathbf{PC}_T \leftarrow \mathbf{PC}_T \setminus X$
18: **end if**
19: **end for**

Algorithm 3. *Hybrid HPC*

Require: $\mathcal{D}$: data set; $\mathbf{U}$: the set of variables
Ensure: A DAG $\mathcal{G}$ on the variables $\mathbf{U}$

1: **for all** pair of nodes $X, Y \in \mathbf{U}$ **do**
2: Add X in $\mathbf{PC}_Y$ and Add Y in $\mathbf{PC}_X$ if $X \in HPC(Y)$ and $Y \in HPC(X)$
3: **end for**
4: Starting from an empty graph, perform greedy hill-climbing with operators *add-edge, delete-edge, reverse-edge.* Only try operator *add-edge* $X \rightarrow Y$ if $Y \in \mathbf{PC}_X$

5 Hybrid HPC (H2PC)

In this section, we discuss the SS phase. The following discussion draws strongly on [35] as the SS phase in Hybrid HPC and MMHC are exactly the same. The idea of constraining the search to improve time-efficiency first appeared in the Sparse Candidate algorithm [12]. It results in efficiency improvements over the (unconstrained) greedy search. All recent hybrid algorithms build on this idea, but employ a sound algorithm for identifying the candidate parent sets. The Hybrid HPC first identifies the parents and children set of each variable, then performs a greedy hill-climbing search in the space of BN. The search begins with an empty graph. The edge addition, deletion, or direction reversal that leads to the largest increase in score (the BDeu score was used) is taken and the search continues in a similar fashion recursively. The important difference from standard greedy search is that the search is constrained to only consider adding an edge if it was discovered by HPC in the first phase. We extend the greedy search with a TABU list [12]. The list keeps the last 100 structures explored. Instead of applying the best local change, the best local change that results in a structure not on the list is performed in an attempt to escape local maxima. When 15 changes occur without an increase in the maximum score ever encountered during search, the algorithm terminates. The overall best scoring structure is then returned. Clearly, the more false positives the heuristic allows to enter candidate PC set, the more computational burden is imposed in the SS phase.

6 Experimental Validation

In this section, we conduct an experimental comparison of H2PC against MMHC on several benchmarks with various data sizes. All the data sets used for the empirical experiments are sampled from eight well-known BNs that have been previously used as benchmarks for BN learning algorithms (see Table 1 for details). We do not claim that those data sets resemble real-world problems, however, they make it possible to compare the outputs of the algorithms with the known structure. All BN benchmarks (structure and probability tables) were downloaded from the *bnlearn* repository[1] [26]. Six sample sizes have been considered:

[1] http://www.bnlearn.com/bnrepository

Table 1. Description of the BN benchmarks used in the experiments

network	# of vars	# of edges	max. degree in/out	domain range	min/med/max PC set size
child	20	25	2/7	2-6	1/2/8
insurance	27	52	3/7	2-5	1/3/9
mildew	35	46	3/3	3-100	1/2/5
alarm	37	46	4/5	2-4	1/2/6
hailfinder	56	66	4/16	2-11	1/1.5/17
munin1	186	273	3/15	2-21	1/3/15
pigs	441	592	2/39	3-3	1/2/41
link	724	1125	3/14	2-4	0/2/17

50, 100, 200, 500, 1500 and 5000. All experiments are repeated 10 times for each sample size and each BN. We investigate the behavior of both algorithms using the same parametric tests as a reference. H2PC was implemented in *R* [31] and integrated into the *bnlearn R* package developed by [26]. The source code of H2PC as well as all data sets used for the empirical tests are publicly available [2]. The threshold considered for the type I error of the test is 0.05. Our experiments were carried out on PC with Intel(R) Core(TM) i5-2520M CPU @2,50 GHz 4Go RAM running under Windows 7 32 bits.

We first investigate the quality of the skeleton returned by H2PC during the CB phase. To this end, we measure the false positive edge ratio, the precision (i.e., the number of true positive edges in the output divided by the number of edges in the output), the recall (i.e., the number of true positive edges divided the true number of edges) and a combination of precision and recall defined as $\sqrt{(1-precision)^2 + (1-recall)^2}$, to measure the Euclidean distance from perfect precision and recall, as proposed in [18]. Second, to assess the quality of the final DAG output at the end of the SS phase, we report the five performance indicators [27] described below:

- the posterior density of the network for the data it was learned from, as a measure of goodness of fit. It is known as the Bayesian Dirichlet equivalent score (BDeu) from [13,7] and has a single parameter, the equivalent sample size, which can be thought of as the size of an imaginary sample supporting the prior distribution. The equivalent sample size was set to 10 as suggested in [16];
- the BIC score [25] of the network for the data it was learned from, again as a measure of goodness of fit;
- the posterior density of the network for a new data set, as a measure of how well the network generalizes to new data;
- the BIC score of the network for a new data set, again as a measure of how well the network generalizes to new data;

[2] http://www710.univ-lyon1.fr/$\sim$aaussem/Software.html

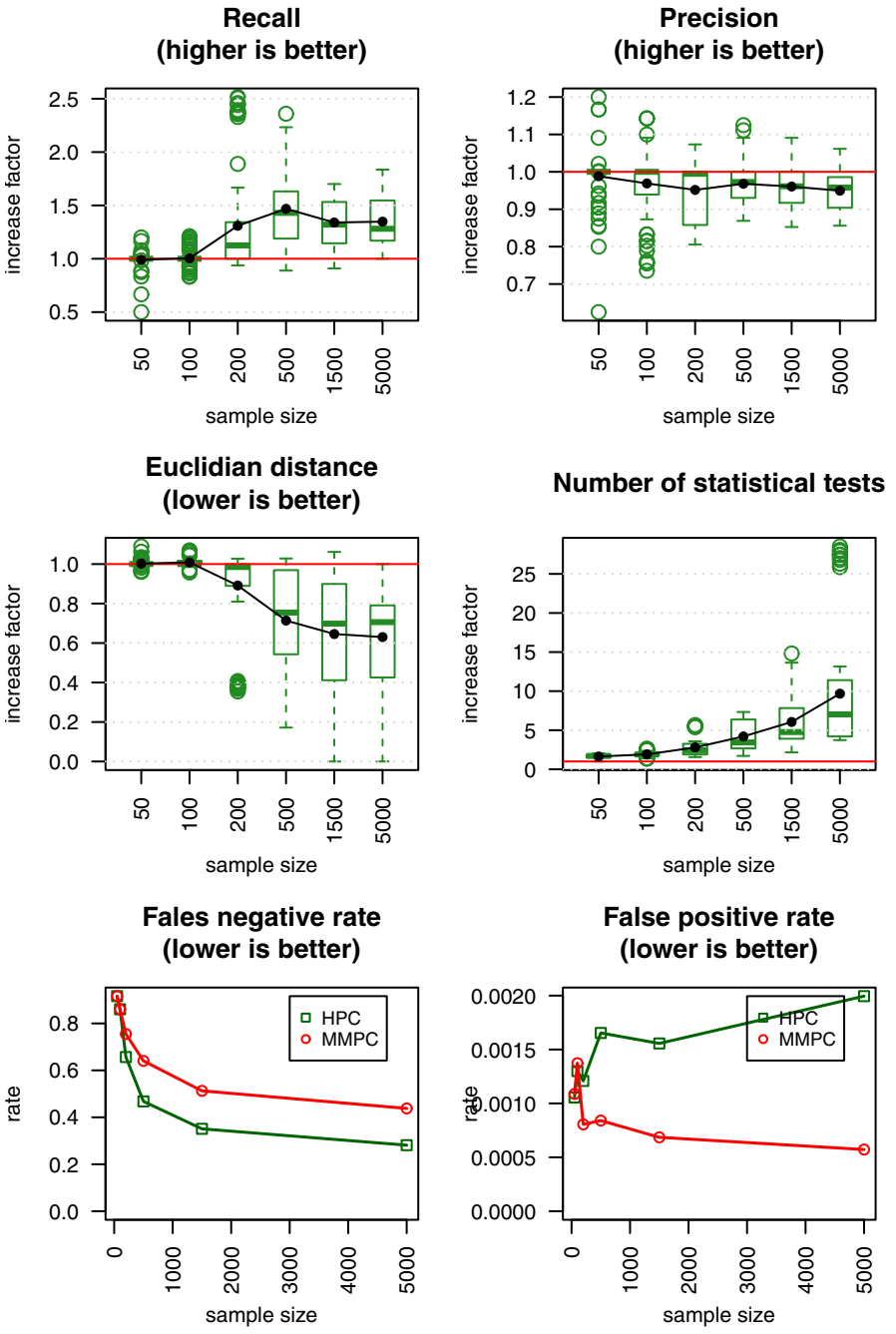


Fig. 1. Quality of the skeleton obtained with HPC over that obtained with MMPC before the SS phase. Results are averaged over the 8 benchmarks.

– the Structural Hamming Distance (SHD) between the learned and the true structure of the network, as a measure of the quality of the learned dependence structure. The SHD between two PDAGs is defined as the number of the following operators required to make the PDAGs match: add or delete an undirected edge, and add, remove, or reverse the orientation of an edge.

For each data set sampled from the true probability distribution of the benchmark, we first learn a network structure with the H2PC and MMHC and then we compute the relevant performance indicators for each pair of network structures. The data set used to assess how well the network generalizes to new data is generated again from the true probability structure of the benchmark networks and contains 5000 observations.

Notice that using the BDeu score as a metric of reconstruction quality has the following two problems. First, the score corresponds to the a posteriori probability of a network only under certain conditions (e.g., a Dirichlet distribution of the hyper parameters); it is unknown to what degree these assumptions hold in distributions encountered in practice. Second, the score is highly sensitive to the equivalent sample size (set to 10 in our experiments) and depends on the network priors used. Since, typically, the same arbitrary value of this parameter is used both during learning and for scoring the learned network, the metric favors algorithms that use the BDeu score for learning. In fact, the BDeu score does not rely on the structure of the original, gold standard network at all; instead it employs several assumptions to score the networks. For those reasons, in addition to the score we also report the BIC score and the SHD metric.

In Figure 1, we report the quality of the skeleton obtained with HPC over that obtained with MMPC (before the SS phase) as a function of the sample size. Results for each benchmark are not shown here in detail due to space restrictions. For sake of conciseness, the performance values are averaged over the 8 benchmarks depicted in Table 1. The increase factor for a given performance indicator is expressed as the ratio of the performance value obtained with HPC over that obtained with MMPC (the gold standard). Note that for some indicators, an increase is actually not an improvement but is worse (e.g., false positive rate, Euclidean distance). For clarity, we mention explicitly on the subplots whether an increase factor > 1 should be interpreted as an improvement or not. Regarding the quality of the superstructure, the advantages of HPC against MMPC are noticeable. As observed, HPC consistently increases the recall and reduces the rate of false negative edges. As expected this benefit comes at a little expense in terms of false positive edges. HPC also improves the Euclidean distance from perfect precision and recall on all benchmarks, while increasing the number of independence tests and thus the running time in the CB phase (see number of statistical tests). It is worth noting that HPC is capable of maintaining the mean false positive edge increase (with respect to MMPC) under $2 \cdot 10^{-3}$ while reducing by 30% the Euclidean distance in the range 500-5000 samples. These results are very much in line with other experiments presented in [23,36].

In Figure 2, we report the quality of the final DAG obtained with H2PC over that obtained with MMHC (after the SS phase) as a function of the sample size.

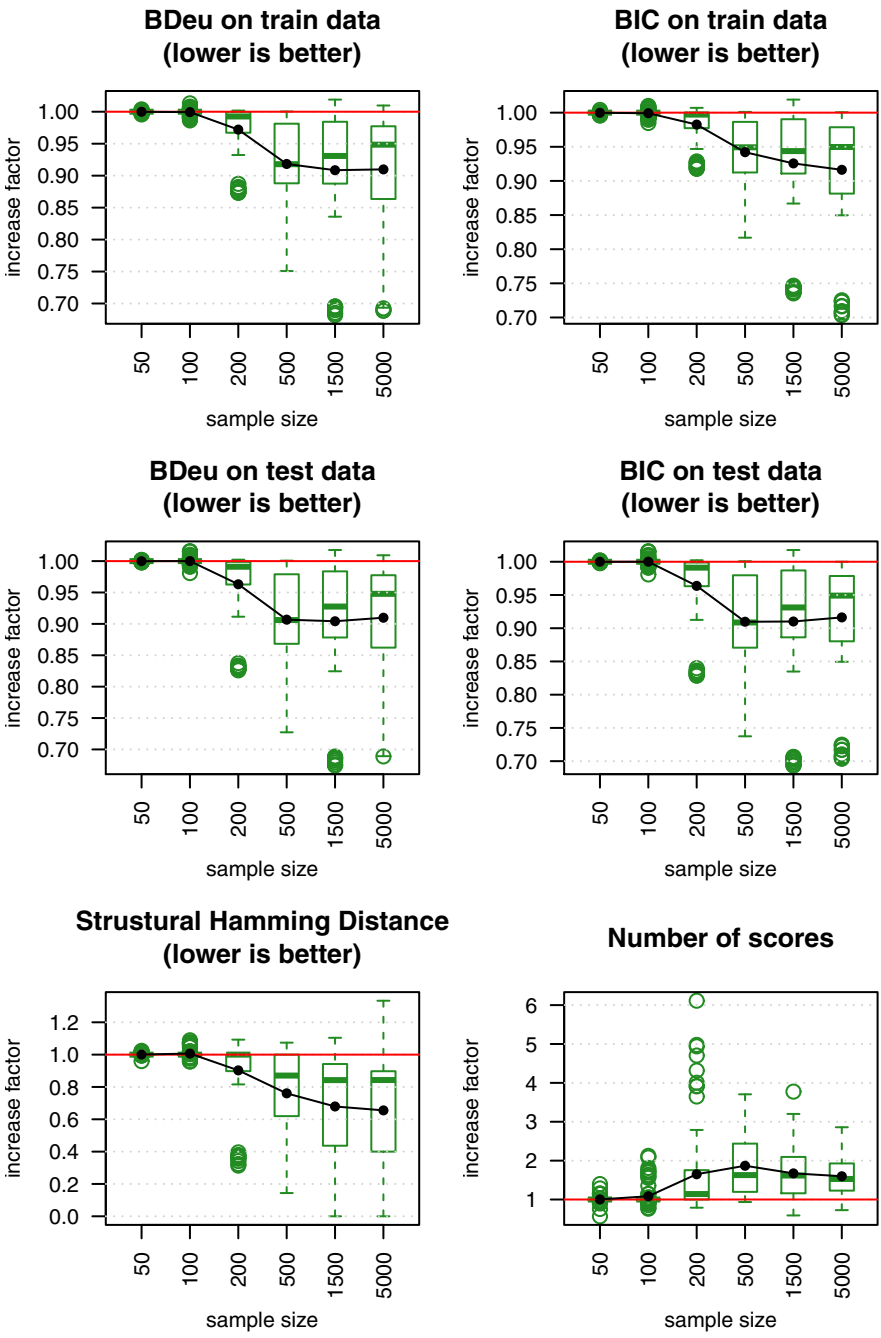


Fig. 2. Quality of the final DAG obtained with H2PC over that obtained with MMHC (after the SS phase). Results are averaged over the 8 benchmarks.

Regarding BDeu and BIC on both training and test data, the improvements are noteworthy. The results in terms of goodness of fit to training data and new data using H2PC clearly dominate those obtained using MMHC, whatever the sample size considered, hence its ability to generalize better. Regarding the quality of the network structure itself (i.e., how close is the DAG to the true dependence structure of the data), this is pretty much a dead heat between the 2 algorithms on small sample sizes (i.e., 50 and 100), however we found H2PC to perform significantly better on larger sample sizes. The SHD increase factor decays rapidly (lower is better) as the sample size increases. As far as the overall running time performance is concerned, we see from Table 2 that both methods have a tendency to work comparatively well for small sample sizes (i.e., less than 200). The total running time with H2PC with 5000 samples is 8 times slower on average than that of MMHC. Overall, it appears that the running time increase factor grows somewhat linearly with the sample size. Nonetheless, it is worth mentioning that our implementation of MMHC in the *bnlearn* package employs several heuristics to speed up learning that are not yet implemented in H2PC. This leads to some loss of efficiency compared to MMHC due to redundant calculations. Notice that the optimization of the HPC code is currently being undertaken to allow for fair comparisons with MMHC.

Overall, H2PC compares favorably to MMHC. It has consistently lower generalization error on all data sets. Large values of the recall do not cause much rise in precision while maintaining the total running time under control. This experiment indicates that MMPC should be best suited in terms of performance when coupled to an optimal SS BN learning method discussed in [22,15].

Table 2. Total running time increase factor (H2PC/MMHC)

Network	Sample Size					
	50	100	200	500	1500	5000
child	1.13 ±0.1	1.28 ±0.2	1.54 ±0.2	2.38 ±0.2	2.65 ±0.2	3.08 ±0.4
insurance	1.21 ±0.2	1.35 ±0.2	2.03 ±0.1	3.83 ±0.2	5.55 ±0.4	7.38 ±0.6
mildew	0.71 ±0.2	1.05 ±0.2	1.10 ±0.1	1.23 ±0.1	1.63 ±0.1	3.19 ±0.3
alarm	1.17 ±0.1	1.39 ±0.1	1.86 ±0.1	2.28 ±0.1	2.93 ±0.3	3.41 ±0.4
hailfinder	1.09 ±0.1	1.30 ±0.1	1.61 ±0.1	2.17 ±0.1	2.82 ±0.2	3.35 ±0.3
munin1	1.09 ±0.0	1.29 ±0.1	1.36 ±0.1	2.01 ±0.1	4.28 ±0.2	12.88 ±0.7
pigs	1.41 ±0.1	1.41 ±0.1	4.65 ±0.2	5.32 ±0.2	6.51 ±0.2	9.70 ±0.3
link	1.57 ±0.0	2.13 ±0.1	2.86 ±0.1	6.07 ±0.2	11.07 ±1.1	23.35 ±0.9
all	1.17 ±0.3	1.40 ±0.3	2.13 ±1.1	3.16 ±1.6	4.68 ±2.9	8.29 ±6.7

7 Discussion

Our prime conclusion is that H2PC is a promising approach to constructing BN structures. The performances of HPC raises interesting possibilities in the context of hybrid methods. It emphasizes that concentrating on higher recall

values while keeping the false positive rate as low as possible pays off in terms of goodness of fit and structure accuracy.

The focus of our study was on the efficiency of the heuristics the learning algorithms are based on, i.e., the maximization algorithms used in score-based algorithms combined with the techniques for learning the dependence structure associated with each node in CB algorithms. The influence of the other components of the overall learning strategy, such as the conditional independence tests (and the associated type I error threshold) or the network scores (and the associated parameters, such as the equivalent sample size), was not investigated.

The conclusions of our studies should be applicable to the shrinkage test that is more robust to small sample sizes, and to the permutation mutual information test for large samples. Tsamardinos et al. [33] recently showed that the use of exact tests based on (semi-parametric) permutation procedures lead to more robust structural learning, while being only 10-20 times slower than the asymptotic tests for small sample sizes. Similarly, Scutari [28] investigated the behavior of permutation conditional independence tests and tests based the permutation Pearson's χ^2 test, the permutation mutual information test, and the shrinkage test based on the estimator for the mutual information. Based on a single BN benchmark, they showed that permutation tests result in better network structures than the corresponding parametric tests in terms of goodness of fit. However, the output graphs are often not as close to the true network structure as the ones learned with the corresponding parametric tests. Shrinkage tests, on the other hand, outperform both parametric and permutation tests in the quality of the network structure itself, which is closer to the true dependence structure but do not fit the data as well as the networks learned with the corresponding maximum likelihood tests. So, there is no clear picture as to which test should be employed on a given data set. This is still an open question.

As noted by [22], it is possible to reduce the complexity of an optimal search of an exponential factor by using a structural constraint such as a super-structure on condition that this super-structure is sound (i.e., includes the true graph as a subgraph). Under this assumption, the accuracy of the resulting graph may greatly improve according. Consequently, more attention should be paid to learning sound superstructures rather than true skeleton from data as both speed and accuracy should be expected with more sophisticated SS search strategies than the greedy HC used in our study. Although sound super-structures are easier to learn for high values of type I error, such values produce denser structures with many extra edges, thereby resulting in high computational overheads. Therefore, relaxing the type I error of the tests is not a solution. [15] show that a small change in the type I error in MMPC yields a dramatical increase of the computational burden involved in their hybrid procedure, with almost no gain in accuracy. The key is to keep the false positive rate small while controlling the false missing rate.

Finally, it is worth mentioning that neither MMPC nor HPC were optimized in this work to learn global superstructures as both MMPC and HPC are run independently on each node without keeping track of the dependencies found

previously. This leads to some loss of efficiency due to redundant calculations. The reason is that they were initially designed to infer a local network around a target node. An optimized version of HPC for super-structure discovery was developed in [36]. The optimizations were done in order to get a global method and to lower the computational cost of HPC, while maintaining its performance. These optimizations include the use of a cache to store the (in)dependencies and the use of a global structure. This optimizations reduce the computational cost of HPC by 30% on average according to the authors.

8 Conclusion

We discussed a hybrid algorithm for BN structure learning called Hybrid HPC (H2PC). Our extensive experiments show that H2PC outperforms MMHC in terms of goodness of fit to training data and new data as well, hence its ability to generalize better, with little overhead in terms of of running time over MMHC. The optimization of the HPC code is currently being undertaken. Regarding the quality of the network structure itself (i.e., how close is the DAG to the true dependence structure of the data), we found H2PC to outperform MMHC by a significant margin. More importantly, our experimental results show a clear benefit in terms of edge recall without sacrificing the number of extra edges, which is crucial for the soundness of the super-structure used during the second stage of hybrid methods like the ones proposed in [22,15]. Though not discussed here, a topic of considerable interest would be to ascertain which independence test is most suited to the data at hand. This needs further substantiation through more experiments and analysis.

Acknowledgments. The authors thank Marco Scutari for sharing his *bnlearn* package in *R*. The experiments reported here were performed on computers funded by a French Institute for Complex Systems (IXXI) grant.

References

1. Agresti, A.: Categorical Data Analysis, 2nd edn. Wiley (2002)
2. Aliferis, C.F., Statnikov, A.R., Tsamardinos, I., Mani, S., Koutsoukos, X.D.: Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. Journal of Machine Learning Research 11, 171–234 (2010)
3. Armen, A.P., Tsamardinos, I.: A unified approach to estimation and control of the false discovery rate in bayesian network skeleton identification. In: European Symposium on Artificial Neural Networks, ESANN 2011 (2011)
4. Aussem, A., Rodrigues de Morais, S., Corbex, M.: Analysis of nasopharyngeal carcinoma risk factors with bayesian networks. Artificial Intelligence in Medicine 54(1) (2012)
5. Aussem, A., Tchernof, A., Rodrigues de Morais, S., Rome, S.: Analysis of lifestyle and metabolic predictors of visceral obesity with bayesian networks. BMC Bioinformatics 11, 487 (2010)

6. Brown, L.E., Tsamardinos, I.: A strategy for making predictions under manipulation. In: JMLR: Workshop and Conference Proceedings, vol. 3, pp. 35–52 (2008)
7. Buntine, W.: Theory refinement on Bayesian networks. In: Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence, San Mateo, CA, USA, pp. 52–60. Morgan Kaufmann Publishers (July 1991)
8. Cawley, G.: Causal and non-causal feature selection for ridge regression. In: JMLR: Workshop and Conference Proceedings vol. 3 (2008)
9. Cheng, J., Greiner, R., Kelly, J., Bell, D.A., Liu, W.: Learning Bayesian networks from data: An information-theory based approach. Artif. Intell. 137(1-2), 43–90 (2002)
10. Chickering, D.M.: Optimal structure identification with greedy search. Journal of Machine Learning Research 3, 507–554 (2002)
11. Ellis, B., Wong, W.H.: Learning causal bayesian network structures from experimental data. Journal of the American Statistical Association 103, 778–789 (2008)
12. Friedman, N.L., Nachman, I., Peér, D.: Learning bayesian network structure from massive datasets: the"sparse candidate" algorithm. In: Laskey, K.B., Prade, H. (eds.) Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, pp. 21–30. Morgan Kaufmann Publishers (1999)
13. Heckerman, D., Geiger, D., Chickering, D.M.: Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning 20(3), 197–243 (1995)
14. Koivisto, M., Sood, K.: Exact bayesian structure discovery in bayesian networks. Journal of Machine Learning Research 5, 549–573 (2004)
15. Kojima, K., Perrier, E., Imoto, S., Miyano, S.: Optimal search on clustered structural constraint for learning bayesian network structure. Journal of Machine Learning Research 11, 285–310 (2010)
16. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
17. Moore, A., Wong, W.-K.: Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In: Fawcett, T., Mishra, N. (eds.) Proceedings of the 20th International Conference on Machine Learning, ICML 2003 (August 2003)
18. Peña, J.M., Nilsson, R., Björkegren, J., Tegnér, J.: Towards scalable and data efficient learning of Markov boundaries. International Journal of Approximate Reasoning 45(2), 211–232 (2007)
19. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
20. Peña, J.M.: Learning Gaussian Graphical Models of Gene Networks with False Discovery Rate Control. In: Marchiori, E., Moore, J.H. (eds.) EvoBIO 2008. LNCS, vol. 4973, pp. 165–176. Springer, Heidelberg (2008)
21. Peña, J.: Finding consensus bayesian network structures. Journal of Artificial Intelligence Research 42, 661–687 (2012)
22. Perrier, E., Imoto, S., Miyano, S.: Finding optimal bayesian network given a superstructure. Journal of Machine Learning Research 9, 2251–2286 (2008)
23. de Morais, S.R., Aussem, A.: An Efficient and Scalable Algorithm for Local Bayesian Network Structure Discovery. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part III. LNCS, vol. 6323, pp. 164–179. Springer, Heidelberg (2010)
24. Rodrigues de Morais, S., Aussem, A.: A novel Markov boundary based feature subset selection algorithm. Neurocomputing 73, 578–584 (2010)

25. Schwarz, G.E.: Estimating the dimension of a model. Journal of Biomedical Informatics 6(2), 461–464 (1978)
26. Scutari, M.: Learning bayesian networks with the bnlearn R package. Journal of Statistical Software 35(3), 1–22 (2010)
27. Scutari, M., Brogini, A.: Bayesian network structure learning with permutation tests. To appear in Communications in Statistics Theory and Methods (2012)
28. Scutari, M.: Measures of Variability for Graphical Models. PhD thesis, School in Statistical Sciences, University of Padova (2011)
29. Silander, T., Myllymaki, P.: Simple approach for finding the globally optimal Bayesian network structure. In: Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI 2006), pp. 445–452 (2006)
30. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search, 2nd edn. The MIT Press (2000)
31. R Development Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2010)
32. Tsamardinos, I., Aliferis, C.F., Statnikov, A.R.: Algorithms for large scale Markov blanket discovery. In: Florida Artificial Intelligence Research Society Conference FLAIRS 2003, pp. 376–381 (2003)
33. Tsamardinos, I., Borboudakis, G.: Permutation Testing Improves Bayesian Network Learning. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part III. LNCS, vol. 6323, pp. 322–337. Springer, Heidelberg (2010)
34. Tsamardinos, I., Brown, L.E.: Bounding the false discovery rate in local Bayesian network learning. In: Proceedings AAAI National Conference on AI AAAI 2008, pp. 1100–1105 (2008)
35. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning 65(1), 31–78 (2006)
36. Villanueva, E., Maciel, C.D.: Optimized algorithm for learning bayesian network superstructures. In: Proceedings of the 2012 International Conference on Pattern Recognition Applications and Methods, ICPRAM 2012 (2012)

Bayesian Network Classifiers with Reduced Precision Parameters

Sebastian Tschiatschek[1], Peter Reinprecht[1],
Manfred Mücke[2,3], and Franz Pernkopf[1]

[1] Signal Processing and Speech Communication Laboratory
Graz University of Technology, Graz, Austria
[2] University of Vienna, Research Group
Theory and Applications of Algorithms, Vienna, Austria
[3] Sustainable Computing Research, Austria
http://www.spsc.tugraz.at

Abstract. Bayesian network classifiers (BNCs) are probabilistic classifiers showing good performance in many applications. They consist of a directed acyclic graph and a set of conditional probabilities associated with the nodes of the graph. These conditional probabilities are also referred to as parameters of the BNCs. According to common belief, these classifiers are insensitive to deviations of the conditional probabilities under certain conditions. The first condition is that these probabilities are not too extreme, i.e. not too close to 0 or 1. The second is that the posterior over the classes is significantly different. In this paper, we investigate the effect of precision reduction of the parameters on the classification performance of BNCs. The probabilities are either determined generatively or discriminatively. Discriminative probabilities are typically more extreme. However, our results indicate that BNCs with discriminatively optimized parameters are almost as robust to precision reduction as BNCs with generatively optimized parameters. Furthermore, even large precision reduction does not decrease classification performance significantly. Our results allow the implementation of BNCs with less computational complexity. This supports application in embedded systems using floating-point numbers with small bit-width. Reduced bit-widths further enable to represent BNCs in the integer domain while maintaining the classification performance.

Keywords: Bayesian Network Classifiers, Custom-precision Analysis, Discriminative Classifiers.

1 Introduction

Pattern recognition is about identifying patterns in input data and assigning labels to this data. Examples of pattern recognition are regression and classification. A classifier has to be learned from a set of training samples by identifying discriminative properties such that new unlabeled samples can be correctly classified. Many approaches and algorithms for this purpose exists. Some of the most

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 74–89, 2012.

competitive approaches are support vector machines [16], neural networks [7] and Bayesian network classifiers (BNCs) [5].

BNCs are probabilistic classifiers that assume a joint probability distribution over the input data and the class labels. They classify new input data as the maximum a-posteriori estimate of the class given this data using the assumed probability distribution. The probability distribution is represented by a Bayesian network (BN). BNs consist of a directed acyclic graph, i.e. the structure, and a set of local conditional probability densities, i.e. the parameters. The classification performance of a BNC is determined by the assumed probability distribution. Finding probability distributions that result in good classifiers is addressed by the tasks of structure [1,5,8,14] and parameter learning [6,8,12,15,16]. Structure learning is not considered in this paper and we assume fixed graph structures. In detail, we consider BNCs with naive Bayes (NB) structures, cf. Figure 1, and tree augmented network structures (TAN) [5].

Parameter learning in BNCs resorts to identifying a probability distribution over the input data and the class labels. This distribution must be compatible with the assumed structure of the BNC. For learning these distributions, we use the maximum likelihood (ML), the maximum conditional likelihood (MCL), and the maximum margin (MM) objectives.

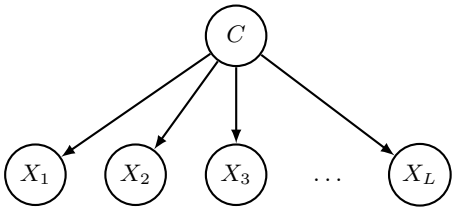


Fig. 1. Naive Bayes structure

The process of parameter learning and classification is typically performed on a computer using high numerical precision, i.e. double-precision floating-point calculations. However, this high precision causes large storage requirements, cf. Table 1. Additionally, the necessary calculations depend on complex computer architectures to be performed efficiently. In contrast to up-to-date computers this requirements are often not met by embedded systems, low energy computers or integrated solutions that need to optimize the used hardware resources. To aid complexity reduction we investigate the performance of BNCs with reduced precision probability parameters. Especially, we are interested in comparing the robustness of generatively (ML) and discriminatively (MCL, MM) optimized probability distributions with respect to precision reduction of their parameters using various BN structures.

Some of our findings can be related to results from sensitivity analysis of BNs [3,4]. Amongst others, the framework of sensitivity analysis describes the

Table 1. Number of probability parameters (# parameters) and the storage requirements (storage) for these parameters in BNCs with different graph structures (for different datasets). Each parameter is assumed to be stored in double-precision floating-point format, i.e. 64 bits are required for each parameter. Details on the structures and datasets are provided in Section 5.

data	structure	# parameters	storage [kB]
USPS	NB	8650	67.6
	TAN-CR	20840	162.8
MNIST	NB	6720	52.5
	TAN-CR	39980	312.3
TIMIT (4 classes)	NB	1320	10.3
TIMIT (6 classes)	NB	1998	15.6

dependency of inference queries to variations in the local conditional probability parameters. The precision reduction of the probability parameters resorts to such variations and can, therefore, be interpreted in this framework. However, the focus in this paper is different. We are particularly interested in analyzing the classification performance of BNCs when reducing the bit-width of all parameters simultaneously. Additionally, we are interested in comparing the robustness of the classification of BNCs with generatively and discriminatively optimized parameters with respect to this precision reduction. As the local conditional probability parameters of discriminatively optimized BNCs tend to be more extreme, we suspected classification rates of these classifiers to depend stronger on the used precision than the classification rates of BNCs with generatively optimized parameters. Nevertheless, our results demonstrate that this is not true.

Our main findings are:

- The number of extreme conditional probability values, i.e. probabilities close to 0 or 1, in BNCs with discriminatively optimized parameters is larger than in BNCs with generatively optimized parameters, cf. Section 5.1. Using results from sensitivity analysis, this suggests that BNCs with discriminatively optimized parameters might be more susceptible to precision reduction than BNCs with generatively optimized parameters. Nevertheless, we observed in experiments that BNCs with both types of parameters can achieve good classification performance using reduced precision floating-point parameters. In fact, the classification performance is close to BNCs with parameters represented in full double-precision floating-point format, cf. Section 5.2.
- The reduction of the precision allows for mapping the classification process of BNCs to the integer domain, cf. Section 4. Thereby, exact computation in that domain, reduced computational complexity and implementation on simple embedded hardware is supported. In fact, some of the considered BNCs can perform classification using integer arithmetic without significant reduction of performance.

The outline of this paper is as follows: In Section 2 we provide a motivating example demonstrating that there is large potential in reducing the precision of the parameters of BNCs. Afterwards, we introduce probabilistic classification, BNCs, and the sensitivity of BNs to changes of their parameters in Section 3. An approach for mapping the parameters of BNCs to the integer domain is presented in Section 4 and various experiments are provided in Section 5. Finally, we conclude the paper in Section 6 and provide a perspective on future work.

2 Motivating Example

In this section we provide an example demonstrating that the parameters of BNs employed for classification do not require high precision. They can be approximated coarsely without reducing the classification rate significantly. In some cases, only a few bits for representing each probability parameter of a BNC are necessary to achieve classification rates close to optimal.

The probability parameters of BNCs, these classifiers are introduced in detail in Section 3, are typical stored in double-precision floating-point format [10, 11]. We use logarithmic probability parameters $w = \log(\theta)$, with $0 \leq \theta \leq 1$, represented as

$$w = (-1)^s \left(1 + \sum_{k=1}^{52} b^k 2^{-k}\right) 2^{\left(\sum_{l=0}^{10} e^l 2^l - 1023\right)}, \tag{1}$$

where $s \in \{0, 1\}$, $b^k \in \{0, 1\}$ for all k, and $e^l \in \{0, 1\}$ for all l. The term

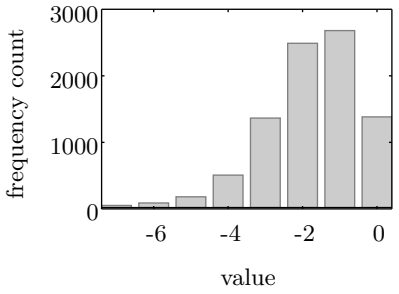
- $(-1)^s$ is the *sign*,
- $(1 + \sum_{i=1}^{52} b^k 2^{-k})$ is the *mantissa*, and
- $(\sum_{l=0}^{10} e^l 2^l - 1023)$ is the *exponent*

of w, respectively. In total 64 bits are used to represent each log-parameter. Processing these parameters on desktop computers does not impose any problems. However, this large bit-width of the parameters can be a limiting factor in embedded systems or applications optimized for low run-times or low energy-consumption.

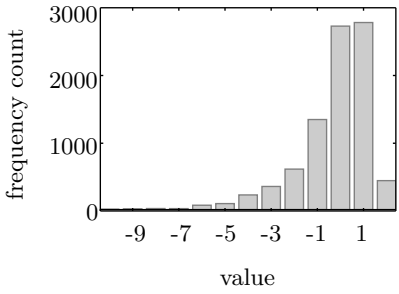
The range of the parameters using double-precision floating-point format is about $\pm 10^{300}$ and by far larger than required; The distribution of the log-parameters of a BNC with maximum likelihood parameters for handwritten digit data (USPS data, details are provided in Section 5) is shown in Figure 2(a). Additionally, the distribution of the values of the exponent is shown in Figure 2(b). All the log-parameters are negative and their range is $[-7; 0]$. The range of the exponent of the logarithmic parameters is $[-10; 2]$.

The required bit-width to store the logarithmic parameters in a floating-point format, cf. Equation (1), can be reduced in three aspects:

1. **Sign bit.** Every probability θ satisfies $0 \leq \theta \leq 1$. Therefore, its logarithm is in the range $-\infty \leq w \leq 0$. Consequently, the sign bit can be removed without any change in the represented parameters.



(a) logarithmic probabilities



(b) exponent of logarithmic conditional probabilities in double-precision

Fig. 2. Histograms of (a) the log-parameters, and (b) the exponents of the log-parameters of a BNC for handwritten digit data with ML parameters assuming NB structure.

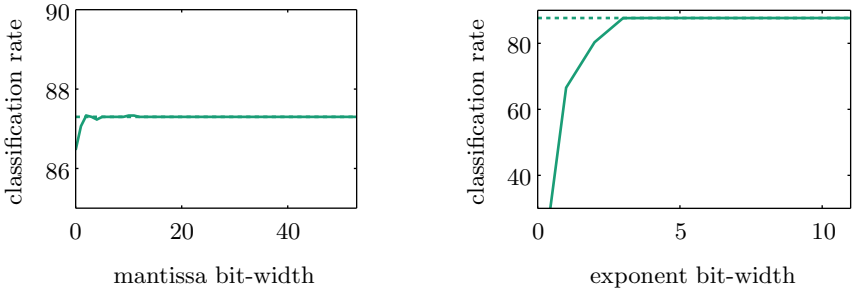
2. **Bit-width of the mantissa.** We varied the bit-width of the mantissa of the log-parameters while keeping the exponent unchanged. As a result, we observed that this does not influence the classification rate significantly when using ML parameters, cf. Figure 3(a). When using 4 or more bits to represent the mantissa, the performance is almost the same as when using the full double-precision floating-point format, i.e. 53 bits for the mantissa.
3. **Bit-width of the exponent.** Changing the bit-width of the exponent has the largest impact on the classification performance. A change of the exponent of a parameter results in a change of the scale of this parameter. The classification rates resulting from reducing the bit-width of the exponent are shown in Figure 3(b). Note that we reduced the bit-width starting with the most significant bit (MSB). Only a few bits are necessary for classification rates on par with the rates achieved using full double-precision floating-point parameters.

Based on this motivating example demonstrating the potential of precision reduction we can even map BNCs to the integer domain, cf. Section 4. Further experimental results are shown in Section 5.

3 Background

3.1 Probabilistic Classification

Probabilistic classifiers are embedded in the framework of probability theory. One assumes a random variable (RV) C denoting the class and RVs $X_1, \ldots, X_L$ representing the attributes/features of the classifier. These RVs are related by a joint probability distribution $\mathrm{P}^*(C, \mathbf{X})$, where $\mathbf{X} = [X_1, \ldots, X_L]$ is a random vector consisting of $X_1, \ldots, X_L$. In typical settings, this joint distribution is unknown and a limited number of samples drawn from true distribution $\mathrm{P}^*(C, \mathbf{X})$,



(a) varying mantissa bit-width, using full bit-width for exponent

(b) varying exponent bit-width, using full bit-width for mantissa

Fig. 3. Classification rate over varying bit-width of (a) the mantissa, and (b) the exponent, for handwritten digit data, NB structure, and log ML parameters. The classification rates using full double-precision logarithmic parameters are indicated by the horizontal dotted lines.

i.e. a training set $\mathcal{D}$, is available. This set $\mathcal{D}$ consists of N i.i.d. labeled samples, i.e. $\mathcal{D} = \{(c^{(n)}, \mathbf{x}^{(n)}) | 1 \leq n \leq N\}$, where $c^{(n)}$ denotes the instantiation of the RV C and $\mathbf{x}^{(n)}$ the instantiation of $\mathbf{X}$ in the n^{th} training sample. The aim is to induce *good* classifiers provided the training set, i.e. classifiers with low generalization error. Formally, a classifiers h is a mapping

$$h: \quad \begin{aligned} &\text{sp}(\mathbf{X}) \rightarrow \text{sp}(C), \\ &\mathbf{x} \mapsto h(\mathbf{x}), \end{aligned} \tag{2}$$

where $\text{sp}(\mathbf{X})$ denotes the set of all assignments of $\mathbf{X}$ and $\text{sp}(C)$ is the set of classes. The generalization error of this classifier is

$$\text{Err}(h) := \mathbb{E}_{\text{P}^*(C,\mathbf{X})} \left[\mathbf{1}\{C \neq h(\mathbf{X})\} \right], \tag{3}$$

where $\mathbf{1}\{A\}$ denotes the indicator function and $\mathbb{E}_{\text{P}^*(C,\mathbf{X})}[\cdot]$ is the expectation operator with respect to the distribution $\text{P}^*(C, \mathbf{X})$. The indicator function $\mathbf{1}\{A\}$ equals one if statement A is true and zero otherwise. Typically, the generalization error can not be evaluated because $\text{P}^*(C, \mathbf{X})$ is unknown but is rather estimated using cross-validation [2].

BNCs with generatively optimized parameters are based on the idea of *approximating* $\text{P}^*(C, \mathbf{X})$ by a distribution $\text{P}^{\mathcal{B}}(C, \mathbf{X})$ and using the induced classifier $h_{\text{P}^{\mathcal{B}}(C,\mathbf{X})}$, given as

$$h_{\text{P}^{\mathcal{B}}(C,\mathbf{X})}: \quad \begin{aligned} &\text{sp}(\mathbf{X}) \rightarrow \text{sp}(C), \\ &\mathbf{x} \mapsto \arg\max_{c \in C} \text{P}^{\mathcal{B}}(C = c | \mathbf{X} = \mathbf{x}), \end{aligned} \tag{4}$$

for classification. In this way, each instantiation $\mathbf{x}$ of $\mathbf{X}$ is classified as the maximum a-posteriori (MAP) estimate of C given $\mathbf{x}$ under $\text{P}^{\mathcal{B}}(C, \mathbf{X})$. BNCs with

discriminatively optimized parameters do not approximate $\mathrm{P}^*(C, \mathbf{X})$ but rather determine $\mathrm{P}^{\mathcal{B}}(C, \mathbf{X})$ such that good classification performance is achieved. Discriminative learning of BNCs is advantageous in cases where the assumed model distribution $\mathrm{P}^{\mathcal{B}}(C, \mathbf{X})$ can not approximate $\mathrm{P}^*(C, \mathbf{X})$ well, for example because of a too limited BN structure. Several approaches for optimizing $\mathrm{P}^{\mathcal{B}}(C, \mathbf{X})$ are discussed in the next section after introducing the concept of Bayesian networks in more detail.

3.2 Bayesian Networks and Learning Bayesian Network Classifiers

Bayesian Networks (BNs) [8,12] are used to represent joint probability distributions in a compact and intuitive way. A BN $\mathcal{B} = (\mathcal{G}, \mathcal{P}_{\mathcal{G}})$ consists of a directed acyclic graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{X_0, \ldots, X_L\}$ is the set of nodes and $\mathbf{E}$ the set of edges of the graph, and a set of local conditional probability distributions $\mathcal{P}_{\mathcal{G}} = \{\mathrm{P}(X_0|Pa(X_0)), \ldots, \mathrm{P}(X_L|Pa(X_L))\}$. The terms $Pa(X_0), \ldots, Pa(X_L)$ denote the set of parents of $X_0, \ldots, X_L$ in $\mathcal{G}$, respectively. We abbreviate the conditional probability $P(X_i = j|Pa(X_i) = \mathbf{h})$ as $\theta^i_{j|\mathbf{h}}$ and the corresponding logarithmic probability as $w^i_{j|\mathbf{h}} = \log(\theta^i_{j|\mathbf{h}})$. Each node of the graph corresponds to an RV and the edges of the graph determine dependencies between these RVs. Throughout this paper, we denote X_0 as C, i.e. X_0 represents the class, and assume that C has no parents in $\mathcal{G}$, i.e. $Pa(C) = \emptyset$. A BN induces a joint probability $\mathrm{P}^{\mathcal{B}}(C, X_1, \ldots, X_L)$ by multiplying the local conditional distributions together, i.e.

$$\mathrm{P}^{\mathcal{B}}(C, X_1, \ldots, X_L) = \mathrm{P}(C) \prod_{i=1}^{L} \mathrm{P}(X_i|Pa(X_i)). \tag{5}$$

BNs for classification can be optimized in two ways: firstly, one can select the graph structure $\mathcal{G}$, and secondly, one can learn the conditional probabilities $\mathcal{P}_{\mathcal{G}}$. Selecting the graph structure is known as structure learning and selecting $\mathcal{P}_{\mathcal{G}}$ is known as parameter learning. The structures considered throughout this paper are fairly simple. In detail, we used naive Bayes structures, cf. Figure 1, and tree augmented network structures (TAN) [5].

For learning the parameters $\mathcal{P}_{\mathcal{G}}$ of a BN two paradigms exist, namely generative parameter learning and discriminative parameter learning:

– In generative parameter learning one aims at identifying parameters representing the generative process that results in the data of the training set. An example of this paradigm is maximum likelihood (ML) learning. Its objective is maximization of the likelihood of the data given the parameters. Formally, ML parameters $\mathcal{P}^{\mathrm{ML}}_{\mathcal{G}}$ are learned as

$$\mathcal{P}^{\mathrm{ML}}_{\mathcal{G}} = \arg\max_{\mathcal{P}_{\mathcal{G}}} \prod_{n=1}^{N} \mathrm{P}^{\mathcal{B}}(c^{(n)}, \mathbf{x}^{(n)}), \tag{6}$$

where $\mathrm{P}^{\mathcal{B}}(C, \mathbf{X})$ is the joint distribution in (5) induced by the BN $(\mathcal{G}, \mathcal{P}_{\mathcal{G}})$.

– In discriminative learning one aims at identifying parameters leading to good classification performance on new samples from $\mathrm{P}^*(C, \mathbf{X})$. Several objectives for this purpose are known in the literature. Throughout this paper, we consider the maximum conditional likelihood (MCL) [15] objective and the maximum margin (MM) [6,13] objective.
MCL parameters $\mathcal{P}_{\mathcal{G}}^{\mathrm{MCL}}$ are obtained as

$$\mathcal{P}_{\mathcal{G}}^{\mathrm{MCL}} = \arg\max_{\mathcal{P}_{\mathcal{G}}} \prod_{n=1}^{N} \mathrm{P}^{\mathcal{B}}(c^{(n)}|\mathbf{x}^{(n)}), \tag{7}$$

where again $\mathrm{P}^{\mathcal{B}}(C, \mathbf{X})$ is the joint distribution induced by the BN $(\mathcal{G}, \mathcal{P}_{\mathcal{G}})$ and $\mathrm{P}^{\mathcal{B}}(C|\mathbf{X})$ denotes the conditional distribution of C given $\mathbf{X}$ determined from $\mathrm{P}^{\mathcal{B}}(C, \mathbf{X})$ as $\mathrm{P}^{\mathcal{B}}(C, \mathbf{X}) = \mathrm{P}^{\mathcal{B}}(C|\mathbf{X}) \cdot \mathrm{P}^{\mathcal{B}}(\mathbf{X})$. Thus, MCL parameters maximize the conditional probability of the class instantiations given the instantiations of the attributes.
MM parameters $\mathcal{P}_{\mathcal{G}}^{\mathrm{MM}}$ are found as

$$\mathcal{P}_{\mathcal{G}}^{\mathrm{MM}} = \arg\max_{\mathcal{P}_{\mathcal{G}}} \prod_{n=1}^{N} \min\left(\gamma, d^{(n)}\right), \tag{8}$$

where $d^{(n)}$ is the margin of the n^{th} sample given as

$$d^{(n)} = \frac{\mathrm{P}^{\mathcal{B}}(c^{(n)}|\mathbf{x}^{(n)})}{\max_{c \neq c^{(n)}} \mathrm{P}^{\mathcal{B}}(c|\mathbf{x}^{(n)})}, \tag{9}$$

and $\gamma > 1$ is a parameter controlling the margin. In this way, the margin *measures* the ratio of the likelihood of the n^{th} sample belonging to the correct class $c^{(n)}$ to belonging to the strongest competing class. The n^{th} sample is correctly classified if $d^{(n)} > 1$ and vice versa.

3.3 Sensitivity of Bayesian Networks

The *sensitivity* of a BN $\mathcal{B} = (\mathcal{G}, \mathcal{P}_{\mathcal{G}})$ describes the change in a query with respect to changes in the local conditional probabilities in $\mathcal{P}_{\mathcal{G}}$. For example, a query is the calculation of a posterior probability of the form $\mathrm{P}^{\mathcal{B}}(\mathbf{X}_q|\mathbf{X}_e)$, with $\mathbf{X}_q, \mathbf{X}_e \subseteq \{C, X_1, \ldots, X_L\}$ and $\mathbf{X}_q \cap \mathbf{X}_e = \emptyset$. Several results on estimating and bounding this sensitivity exist in the literature, cf. for example [3,17]. The results therein essentially state that the sensitivity of BNs depends mainly on probability parameters being close to 0 or 1 and queries being close to uniform.

In this context, consider the following theorem:

Theorem 1 (from [3]). *Let X_i be a binary RV in a BN $\mathcal{B} = (\mathcal{G}, \mathcal{P}_{\mathcal{G}})$, then*

$$\left| \frac{\partial \mathrm{P}^{\mathcal{B}}(X_i|\mathbf{X}_e)}{\partial \tau_{X_i|Pa(X_i)}} \right| \leq \frac{\mathrm{P}^{\mathcal{B}}(X_i|\mathbf{X}_e) \cdot (1 - \mathrm{P}^{\mathcal{B}}(X_i|\mathbf{X}_e))}{\mathrm{P}^{\mathcal{B}}(X_i|Pa(X_i)) \cdot (1 - \mathrm{P}^{\mathcal{B}}(X_i|Pa(X_i)))}, \tag{10}$$

where $\tau_{X_i|Pa(X_i)}$ *is a meta-parameter such that* $\mathrm{P}^{\mathcal{B}}(X_i = 0|Pa(X_i)) = \tau_{X_i|Pa(X_i)}$ *and* $\mathrm{P}^{\mathcal{B}}(X_i = 1|Pa(X_i)) = 1 - \tau_{X_i|Pa(X_i)}$.

The theorem states that the magnitude of the partial derivative of $\mathrm{P}^{\mathcal{B}}(X_i|\mathbf{X}_e)$ with respect to $\tau_{X_i|Pa(X_i)}$ is bounded above. The bound depends on the query under the current parameters $\mathrm{P}^{\mathcal{B}}(X_i|\mathbf{X}_e)$ and on the conditional probabilities $\mathrm{P}^{\mathcal{B}}(X_i|Pa(X_i))$. The partial derivative is large whenever $\mathrm{P}^{\mathcal{B}}(X_i|\mathbf{X}_e)$ is close to uniform and whenever $\mathrm{P}^{\mathcal{B}}(X_i = 0|Pa(X_i))$ is close to 0 or 1. In classification the query of interest is the probability of the class variable given the features, i.e. $\mathrm{P}^{\mathcal{B}}(X_i|\mathbf{X}_e) = \mathrm{P}^{\mathcal{B}}(C|\mathbf{X})$. Discriminative objectives for parameter learning in BNs aim at good class separation, i.e. $\mathrm{P}^{\mathcal{B}}(C|\mathbf{X})$ or $1 - \mathrm{P}^{\mathcal{B}}(C|\mathbf{X})$ is typically large. However, also the parameters tend to be extreme, i.e. $\mathrm{P}^{\mathcal{B}}(X_i|Pa(X_i))$ is close to 0 or 1 (some empirical results supporting this are shown in Section 5.1). We expect the bound to be large for discriminatively optimized parameters, as the denominator in the above theorem scales the bound inversely proportional [3]. Hence, either the bound is loose or the partial derivative is actually large resulting in high sensitivity to parameter deviations. This could be the tripping hazard for BNCs with discriminatively optimized parameters. However, experimental observations in Section 5.2 show a robust classification behavior using discriminatively optimized small bit-width parameters.

The above Theorem only describes the sensitivity with respect to a single parameter. There are some extensions of sensitivity analysis describing the sensitivity of queries with respect to changes of many parameters [4]. However, to the best of the authors knowledge, these do not extend to changes of all parameters, which is the focus of this paper. Furthermore, in classification we are not directly interested in the sensitivity of certain queries. The focus is rather on the maximum of a set of queries, i.e. the sensitivity of the MAP classification. Further analytical analysis is intended for future work.

4 BNCs in the Integer Domain

In this section we present how to cast classification using BNCs to the integer domain. This is possible when using reduced precision log-parameters for the BNCs. Without reduced precision, the mapping can not be achieved considering the large range of numbers representable by double-precision floating-point numbers.

Remember, a BNC given by the BN $\mathcal{B} = (\mathcal{G}, \mathcal{P}_{\mathcal{G}})$ assigns an instantiation $\mathbf{x}$ of the attributes to class

$$c = \arg\max_{c' \in \mathrm{sp}(C)} \mathrm{P}^{\mathcal{B}}(c', \mathbf{x}) \tag{11}$$

$$= \arg\max_{c' \in \mathrm{sp}(C)} \mathrm{P}(C = c') \prod_{i=1}^{L} \mathrm{P}(X_i = \mathbf{x}(X_i)|Pa(X_i) = \mathbf{x}(Pa(X_i))), \tag{12}$$

where $\mathbf{x}(X_k)$ denotes the entry in $\mathbf{x}$ corresponding to X_k. This classification rule can be equivalently stated in the logarithmic domain, i.e. $\mathbf{x}$ is assigned to class

$$c = \arg \max_{c' \in \mathrm{sp}(C)} \left[\log \mathrm{P}(C = c') + \sum_{i=1}^{L} \log \mathrm{P}(X_i = \mathbf{x}(X_i) | Pa(X_i) = \mathbf{x}(Pa(X_i))) \right]. \quad (13)$$

As shown in Sections 2 and 5 the logarithmic probabilities in the above equation can often be represented using only a few bits without reducing the classification rate significantly. In many cases, 2 bits for the mantissa and 4 bits for the exponent are sufficient to achieve good classification rates. Using these 6 bits, the logarithmic probability $w^i_{j|\mathbf{h}} = \log \theta^i_{j|\mathbf{h}}$ is given as

$$w^i_{j|\mathbf{h}} = -(1 + b^{i,1}_{j|\mathbf{h}} \cdot 2^{-1} + b^{i,2}_{j|\mathbf{h}} \cdot 2^{-2}) \cdot 2^{\left(\sum_{k=0}^{3} e^{i,k}_{j|\mathbf{h}} \cdot 2^k - 7\right)}. \quad (14)$$

Hence,

$$c = \arg \max_{c' \in \mathrm{sp}(C)} \left[w^0_{c'} + \sum_{i=1}^{L} w^i_{\mathbf{x}(X_i)|\mathbf{x}(Pa(X_i))} \right] \quad (15)$$

$$= \arg \min_{c' \in \mathrm{sp}(C)} \left[-w^0_{c'} - \sum_{i=1}^{L} w^i_{\mathbf{x}(X_i)|\mathbf{x}(Pa(X_i))} \right] \quad (16)$$

$$= \arg \min_{c' \in \mathrm{sp}(C)} \Bigg[(1 + b^{0,1}_{c'} \cdot 2^{-1} + b^{0,2}_{c'} \cdot 2^{-2}) \cdot 2^{\left(\sum_{k=0}^{3} e^{i,k}_{c'} \cdot 2^k - 7\right)} + \quad (17)$$

$$\sum_{i=1}^{L} (1 + b^{i,1}_{\mathbf{x}(X_i)|\mathbf{x}(Pa(X_i))} 2^{-1} + b^{i,2}_{\mathbf{x}(X_i)|\mathbf{x}(Pa(X_i))} 2^{-2}) \cdot 2^{\left(\sum_{k=0}^{3} e^{i,k}_{\mathbf{x}(X_i)|\mathbf{x}(Pa(X_i))} 2^k - 7\right)} \Bigg].$$

Multiplying (17) by the constant 2^9 does not change the classification. Hence, classification can be performed by

$$c = \arg \min_{c' \in \mathrm{sp}(C)} \Bigg[(4 + b^{0,1}_{c'} \cdot 2 + b^{0,2}_{c'}) \cdot 2^{\left(\sum_{k=0}^{3} e^{i,k}_{c'} \cdot 2^k\right)} + \quad (18)$$

$$\sum_{i=1}^{L} (4 + b^{i,1}_{\mathbf{x}(X_i)|\mathbf{x}(Pa(X_i))} \cdot 2 + b^{i,2}_{\mathbf{x}(X_i)|\mathbf{x}(Pa(X_i))}) \cdot 2^{\left(\sum_{k=0}^{3} e^{i,k}_{\mathbf{x}(X_i)|\mathbf{x}(Pa(X_i))} \cdot 2^k\right)} \Bigg]$$

which resorts to integer computations only. Furthermore, no floating-point rounding errors of any kind are introduced during computation when working purely in the integer domain. Integer arithmetic is sufficient for implementation.

5 Experiments

In this section we present classification experiments using reduced precision log probability parameters of BNCs. Throughout this section we consider the following three datasets:

- **TIMIT-4/6 Data.** This dataset is extracted from the TIMIT speech corpus using the dialect speaking region 4. It consists of 320 utterances from 16 male and 16 female speakers. Speech frames are classified into either four or six classes using 110134 and 121629 samples, respectively. Each sample is represented by 20 mel-frequency cepstral coefficients (MFCCs) and wavelet-based features [13]. We perform classification experiments on data of both genders (Ma+Fe).
- **USPS Data.** This dataset contains 11000 uniformly distributed handwritten digit images from zip codes of mail envelopes. Each digit is represented as a 16×16 grayscale image, where each pixel is considered as feature.
- **MNIST Data [9].** This dataset contains 70000 samples of handwritten digits. The digits represented by gray-level images were down-sampled by a factor of two resulting in a resolution of 16×16 pixels, i.e. 196 features.

Some of the experiments are performed using different BN structures. In detail, we considered the naive Bayes (NB) structure, the generative TAN-CMI structure [5] and the discriminative TAN-OMI-CR and TAN-CR structures [14]. The discriminative structures are determined by search-and-score heuristics using the classification rate (CR) as score.

5.1 Number of Extreme Parameter Values in BNCs

We determined BNCs with ML, MCL and MM parameters. For calculating the MCL and MM parameters we used the conjugate gradient based approaches proposed in [13]. However, we did not use the proposed early-stopping heuristic for determining the number of conjugate gradient iterations but rather performed up to 200 iterations (or until there was no further increase in the objective). We then counted the number of conditional probability parameters with a maximal distance of ϵ to the extreme values 0 and 1, i.e. the count is given as

$$M_\epsilon = \sum_{i,j,\mathbf{h}} \mathbf{1}\{(1 - \theta^j_{j|\mathbf{h}}) < \epsilon\} + \sum_{i,j,\mathbf{h}} \mathbf{1}\{\theta^j_{j|\mathbf{h}} < \epsilon\}. \tag{19}$$

The results for USPS and MNIST data are shown in Tables 2(a) and 2(b), respectively. The number of extreme parameter values in BNCs with MCL parameters is larger than in BNCs with MM parameters, and the number of extreme parameter values in BNCs with MM parameters is larger than in BNCs with ML parameters. This suggests that classification using MCL parameters is more sensitive to parameter deviations than classification with MM parameters, and classification using MM parameters is more sensitive to deviations than classification with ML parameters.

5.2 Reduced Precision Classification Performance

We evaluated the classification performance of BNCs with ML, MCL and MM parameters on the USPS, MNIST and TIMIT data. Results are shown in

Table 2. Number of probability parameters $\theta^i_{j|\mathbf{h}}$ close to the extreme values 0 and 1. Additionally, the total number of parameters (# par.) and classification rates (CR) on the test set using parameters in full double-precision floating-point format on (a) USPS data and (b) MNIST data are shown.

(a) USPS

		$M_{0.05}$			$M_{0.01}$			CR		
structure	# par.	ML	MCL	MM	ML	MCL	MM	ML	MCL	MM
NB	8650	1478	4143	1837	364	2134	446	87.10	93.93	95.00
TAN-CMI	33040	12418	14712	13002	8271	9371	8428	91.90	95.70	95.37
TAN-OMI-CR	25380	6677	8167	7441	3486	3937	3624	92.40	95.73	95.40
TAN-CR	20840	5405	7344	6519	2666	3503	3009	92.57	95.97	95.87

(b) MNIST

		$M_{0.05}$			$M_{0.01}$			CR		
structure	# par.	ML	MCL	MM	ML	MCL	MM	ML	MCL	MM
NB	6720	3252	3289	3170	1784	1513	1520	83.73	92.00	91.97
TAN-CMI	38350	15772	25327	16790	8603	18647	9448	91.28	92.91	94.21
TAN-OMI-CR	44600	22488	29159	24048	13615	20419	15147	92.01	93.59	94.60
TAN-CR	39980	19557	25733	23308	11794	17702	16020	92.58	93.72	95.02

Figures 4, 5, and 6, respectively. Classification rates using full double-precision floating-point parameters are indicated by the dotted lines. The classification performance resulting from BNCs with reduced precision ML, MCL, and MM parameters are shown by the solid lines. Reduced precision parameters were determined by firstly learning parameters in double-precision, and secondly reducing the precision of these parameters. Even when using only 4 bits to represent the exponent and 1 bit to represent the mantissa, the classification rates are close to full-precision performance on USPS data. On MNIST and TIMIT data the results are similar when 4 and 2 bits are used to represent the mantissa, respectively.

Furthermore, we evaluated the classification performance of BNCs with reduced precision parameters using a varying size of the training set. The training sets were obtained by selecting the desired number of samples randomly from all available samples. The remaining samples were used as test set. For every sample size, 5 different training/test splits were evaluated. Results on USPS data are shown in Figure 7. Classification performance using reduced precision parameters is close to optimal for all sample sizes.

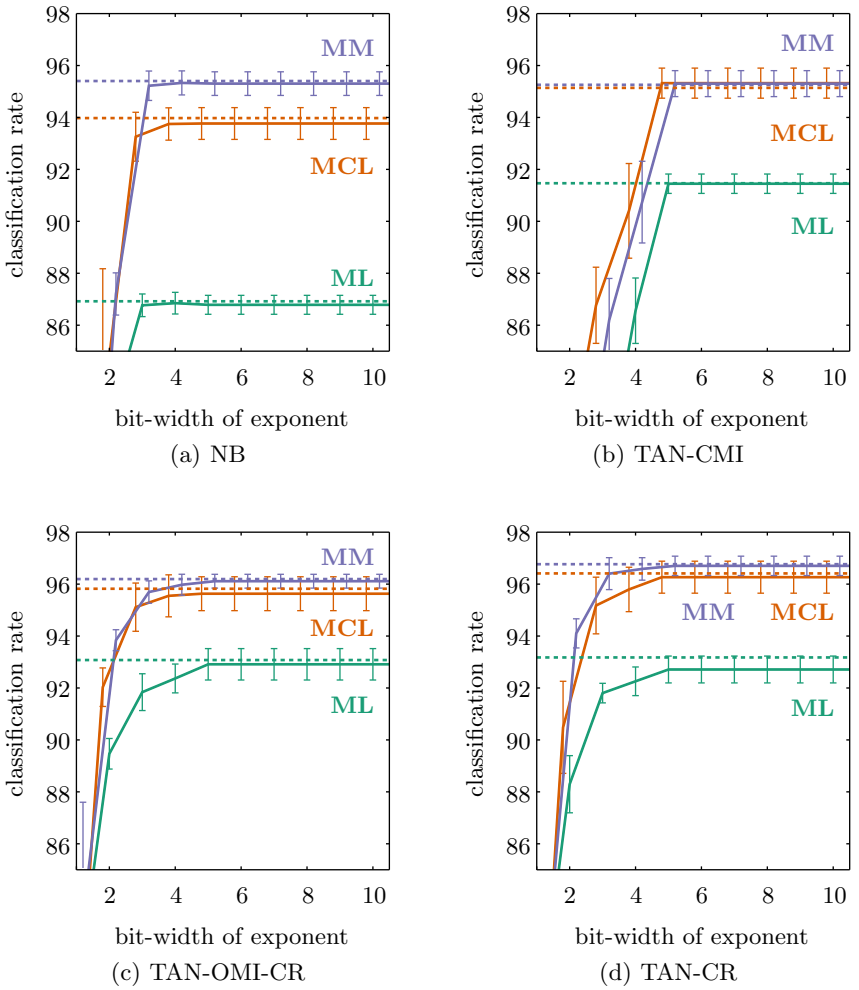


Fig. 4. Classification rates of BNCs with (a) NB, (b) TAN-CMI, (c) TAN-OMI-CR, and (d) TAN-CR structures using reduced precision ML, MCL, and MM parameters on USPS data. The bit-width of the mantissa was fixed to 1 bit and the bit-width of the exponent was varied. The classification rates for full double-precision floating-point parameters are indicated by the horizontal dotted lines. Error bars indicate the 95 % confidence intervals of the mean classification rate over 5 different training/test splits.

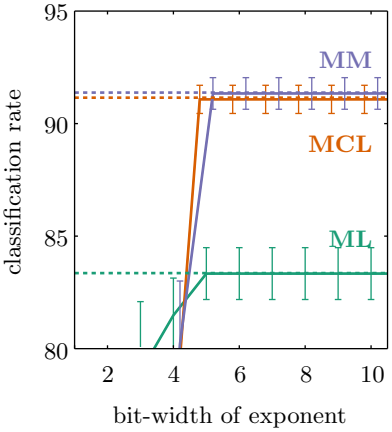


Fig. 5. Classification rates of BNCs with NB structure using reduced precision ML, MCL, and MM parameters on MNIST data. The bit-width of the mantissa was fixed to 4 bits and the bit-width of the exponent was varied. The classification rate for full double-precision floating-point parameters is indicated by the horizontal dotted lines. Error bars indicate the 95 % confidence intervals of the mean classification rate over 5 different training/test splits.

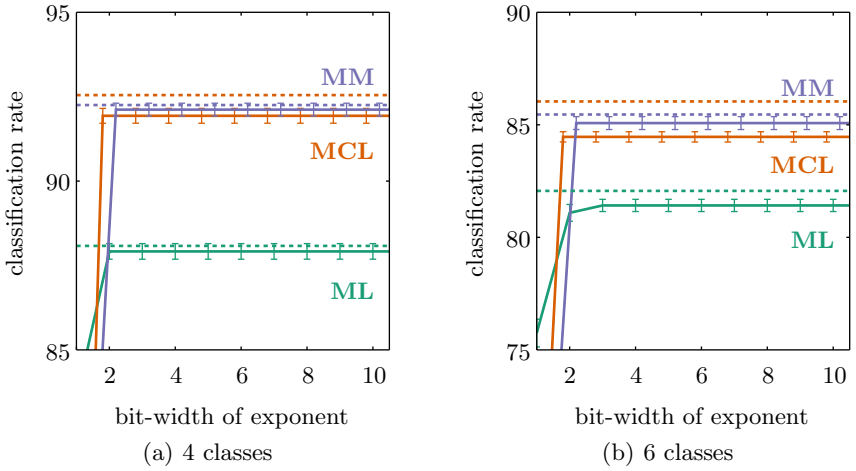
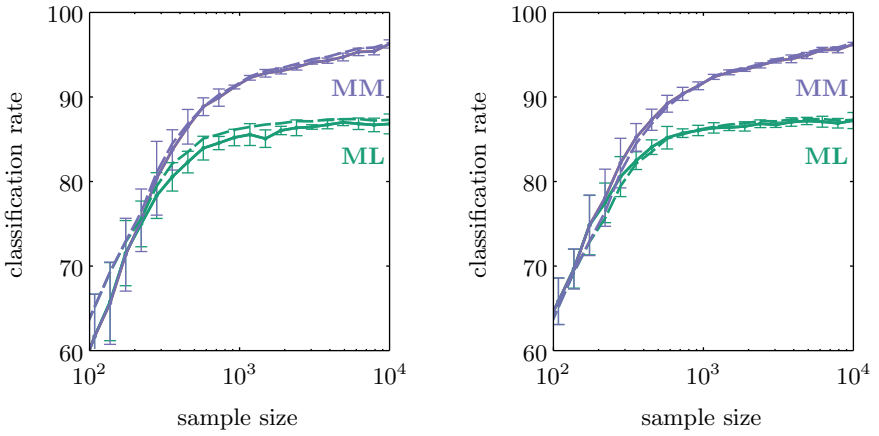


Fig. 6. Classification rates of BNCs with NB structure using ML, MCL, and MM parameters with reduced precision on TIMIT data with (a) 4 classes and (b) 6 classes. The bit-width of the mantissa was fixed to 2 bits and the bit-width of the exponent was varied. The classification rates for full double-precision floating-point parameters are indicated by the horizontal dotted lines. Error bars indicate the 95 % confidence intervals of the mean classification rate over 5 different training/test splits.



(a) 3 bit exponent bit-width (b) 5 bit exponent bit-width

Fig. 7. Classification rates of BNCs with NB structures using reduced precision ML and MM parameters on USPS data. The parameters were learned from training sets with varying sizes. The bit-width of the mantissa was fixed to 1 bit. The bit-width of the exponent is 3 bits in (a) and 5 bits in (b). The classification rates for full double-precision floating-point parameters using the same training data are indicated by the dashed lines. Error bars indicate the 95 % confidence intervals of the mean classification rate over 5 different training/test splits.

6 Conclusion and Further Work

In this paper, we presented classification results of BNCs when reducing the precision of the probability parameters. Contrary to the authors' expectation, even discriminatively optimized BNCs are robust to distortions in the parameters resulting from the bit-width reduction. About 6 to 10 bits are necessary to represent each probability parameter while maintaining classification rates close to full-precision performance. This allows either to implement BNCs with reduced precision floating point arithmetic or to cast the classification to the integer domain. In both cases, computational and run-time benefits arise when implementing BNCs on embedded systems or low-power computers.

Future work aims to address the following issues:

1. Analytical determination of the minimum bit-width of the probability parameters of BNCs such that classification rates are close to full-precision performance. Results from sensitivity analysis are to be used. The analysis will be performed for different datasets and classifier structures.
2. Implementation of BNCs in the integer domain and measuring the computational complexity reduction.

Acknowledgments. This work was supported by the Austrian Science Fund (project numbers P22488-N23, S10608-N13 and S10610-N13).

References

1. Acid, S., Campos, L.M., Castellano, J.G.: Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. Machine Learning 59, 213–235 (2005)
2. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer (2007)
3. Chan, H., Darwiche, A.: When do numbers really matter? Artificial Intelligence Research 17(1), 265–287 (2002)
4. Chan, H., Darwiche, A.: Sensitivity analysis in Bayesian networks: From single to multiple parameters. In: Uncertainty in Artificial Intelligence (UAI), pp. 67–75 (2004)
5. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning, 131–163 (1997)
6. Guo, Y., Wilkinson, D., Schuurmans, D.: Maximum margin Bayesian networks. In: Uncertainty in Artificial Intelligence (UAI), pp. 233–242 (2005)
7. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, Upper Saddle River (1998)
8. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
9. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
10. Muller, J.M., Brisebarre, N., de Dinechin, F., Jeannerod, C.P., Lefèvre, V., Melquiond, G., Revol, N., Stehlé, D., Torres, S.: Handbook of Floating-Point Arithmetic. Birkhäuser Boston (2010)
11. Overton, M.L.: Numerical computing with IEEE floating point arithmetic - including one theorem, one rule of thumb, and one hundred and one exercices. Society for Industrial and Applied Mathematics (SIAM) (2001)
12. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
13. Pernkopf, F., Wohlmayr, M., Tschiatschek, S.: Maximum margin Bayesian network classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 34(3), 521–531 (2012)
14. Pernkopf, F., Bilmes, J.A.: Efficient heuristics for discriminative structure learning of Bayesian network classifiers. Journal of Machine Learning Research (JMLR) 11, 2323–2360 (2010)
15. Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., Tirri, H.: On discriminative Bayesian network classifiers and logistic regression. Machine Learning 59(3), 267–296 (2005)
16. Vapnik, V.N.: Statistical Learning Theory. Wiley (1998)
17. Wang, H.: Using sensitivity analysis for selective parameter update in Bayesian network learning. In: Association for the Advancement of Artificial Intelligence, AAAI (2002)

Combining Subjective Probabilities and Data in Training Markov Logic Networks

Tivadar Pápai[1], Shalini Ghosh[2], and Henry Kautz[1]

[1] Department of Computer Science,University of Rochester, Rochester, NY
{papai,kautz}@cs.rochester.edu
[2] Computer Science Laboratory, SRI International, Menlo Park, CA
shalini@csl.sri.com

Abstract. Markov logic is a rich language that allows one to specify a knowledge base as a set of weighted first-order logic formulas, and to define a probability distribution over truth assignments to ground atoms using this knowledge base. Usually, the weight of a formula cannot be related to the probability of the formula without taking into account the weights of the other formulas. In general, this is not an issue, since the weights are learned from training data. However, in many domains (*e.*g. healthcare, dependable systems, *e*tc.), only little or no training data may be available, but one has access to a domain expert whose knowledge is available in the form of subjective probabilities. Within the framework of Bayesian statistics, we present a formalism for using a domain expert's knowledge for weight learning. Our approach defines priors that are different from and more general than previously used Gaussian priors over weights. We show how one can learn weights in an MLN by combining subjective probabilities and training data, without requiring that the domain expert provides consistent knowledge. Additionally, we also provide a formalism for capturing conditional subjective probabilities, which are often easier to obtain and more reliable than non-conditional probabilities. We demonstrate the effectiveness of our approach by extensive experiments in a domain that models failure dependencies in a cyber-physical system. Moreover, we demonstrate the advantages of using our proposed prior over that of using non-zero mean Gaussian priors in a commonly cited social network MLN testbed.

1 Introduction

Markov logic [1], a language widely used for relational learning, represents knowledge by a set of weighted first-order logic formulas. However, except for Markov Logic Networks (MLNs) with special structure, the weights cannot be interpreted as probabilities or simple functions of probabilities. The probability of a particular weighted formula can only be computed by taking into account all of the weights in all of the formulas in the full grounding of the MLN. When weights are learned from training data without any prior knowledge, the non-informative nature of individual weights is not problematic. However, in many domains, one may have little or no training data, but instead have access to a domain expert's

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 90–105, 2012.

subjective probabilities and *subjective conditional probabilities*. For example, in the healthcare domain, one might have little data about certain rare diseases, but a doctor may have a subjective notion of what percentage of a rare disease occurs among her patients. Another domain we consider in our experiments models fault-tolerant systems. There may be a paucity of failure data, but an engineer could supply subjective conditional probabilities such as, "If the failure probabilities of the individual components of a system are of the order of 10^{-3}, then the overall system failure probability is of the order of 10^{-4}". In this paper, we provide a formal account of how such domain knowledge can be incorporated into an MLN. Our approach applies to arbitrary MLNs, and in particular, is not restricted to the known special cases of MLNs whose structure corresponds to Bayesian Networks or *chordal graphs* (which are discussed in more detail below). We describe two approaches for encoding domain knowledge as *priors*: the first requires the expert's knowledge to be a consistent set of non-conditional probabilities, while the second, more general, approach allows inconsistent knowledge and conditional probabilities, but has a non-convex optimization subproblem.

We also demonstrate that earlier approaches to incorporating knowledge by defining non-zero mean Gaussian priors over the weights of a MLN (*e.g.*, as implemented in Alchemy [2]) can only be justified in MLNs with special structure, and even then they have certain disadvantages compared to our approach.

The rest of the paper is organized as follows: Sec. 2 covers the mathematical background; Sec. 3 shows the connection between the expected values of features of MLNs and the subjective probabilities provided by an expert; Sec. 4 discusses the disadvantages of using Gaussian priors on the weights of an MLN; Secs. 5 and 6 define the two different type of priors we investigate; Sec. 7 describes our experiments; and Secs. 8 and 9 discuss related work, summarize our results, and lay out our planned future work.

2 Background

2.1 Markov Logic Network

Markov logic [1] is a knowledge representation language that uses weighted formulas in first-order logic to compactly encode probability distributions over relational domains. A *Markov logic network* is a set of weighted first-order logic formulas and a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$ which together define a Markov network $M_{L,C}$ which contains a binary node for each possible grounding of each predicate (ground atom) and a binary valued feature for each grounding of each first-order logic formula. In each truth assignment to the ground atoms, the value of a node is 1 if the corresponding ground predicate is *true*, and 0 otherwise. Similarly, the value of a feature is 1 if the corresponding ground formula is *true*, and 0 otherwise. In this paper we assume function-free clauses and Herbrand interpretations. The probability of a truth assignment (world) x to the ground atoms in an MLN is defined as:

$$Pr(X = x|w) = \frac{\exp(\sum_i w_i n_i(x))}{Z(w)}, \tag{1}$$

where $n_i(x)$ is the number of true groundings of the i-th formula, w_i is the weight of the i-th formula and Z is the normalization factor. We sometimes refer to ground atoms as (random) variables, but these are not to be confused with the quantified variables (ranging over C) that appear in first-order formulas.

2.2 Exponential Families of Probability Distributions

The probability distributions defined by Markov Logic Networks belong to the exponential families of distributions [3], since (1) can be rewritten in a more general form:

$$\Pr(X = x) = \exp\left(\langle \theta, f(x) \rangle - A(\theta)\right) , \tag{2}$$

where θ_i are the *natural parameters* of the distribution, f_i are the features, and $A(\theta)$ is responsible for the normalization. As one can tell by comparing (1) and (2), θ corresponds to w, f_i to n_i and $A(\theta) = \log Z(w)$. The probability (likelihood) of training data $\mathcal{D} = \{x_1, \ldots, x_N\}$ is (with the usual *i.i.d.* assumption):

$$\Pr(\mathcal{D}) = \exp\left(\left\langle \theta, \sum_{d=1}^{N} f(x_d) \right\rangle - N \cdot A(\theta)\right) , \tag{3}$$

As we can see, (3) depends upon the data only through $\sum_{d=1}^{N} f(x_d)$ (a *sufficient statistic* of the data set $\mathcal{D}$). Let $f(\mathcal{D}) = \sum_{d=1}^{N} f(x_d)$ and $\overline{f}(\mathcal{D}) = \frac{1}{N} \sum_{d=1}^{N} f(x_d)$. θ is usually set to maximize (3) for the given training data.

In (2) the distribution is parameterized by its natural parameters (θ). However, it can also be parameterized by its *mean parameters*, where the means are defined as the expected values of the features:

$$\mu_i = \sum_x f_i(x) Pr(X = x) = \mathbb{E}\left[f_i\right] . \tag{4}$$

There is a many-to-one mapping from θ to μ. We use $\theta_{F(x)}$ and $\mu_{F(x)}$ to denote the component of the vectors corresponding to the feature representing the true groundings of formula $F(x)$, and in general follow this convention for vectors of parameters. We will use the notation $\mu(\theta)$ when we want to emphasize the dependence of μ on θ. Since either of μ or θ completely determine the distribution, we have the choice of defining a prior either over μ or θ to represent the knowledge of the expert. The prior we will define over θ restricts the kind of subjective probabilities the expert provides, but will result in a convex optimization problem, thereby making the approach computationally attractive. On the other hand, the prior we will define over μ is less restrictive, allowing both conditional and inconsistent probabilistic constraints, at the worst-case cost of requiring the solution of a non-convex optimization problem. However, we will also discuss special cases when the optimization problem can be solved by simple gradient descent, or at least guarantees can be given for the quality of the result found at any point where the gradient becomes zero.

3 Relationship between Subjective Probabilities and the Parameters of the Exponential Family

We consider the case where a domain expert provides subjective probabilities for some or all of the formulas in an MLN. For example, if $F(x)$ is a formula where x is a vector of (implicitly) universally-quantified variables, the expert can estimate how likely it is that a randomly chosen grounding of $F(x)$ is true. The expert may also provide subjective conditional probabilities over ground formulas. For example, if $F_1(x_1)$ and $F_2(x_2)$ are formulas, then for chosen groundings c_1 and c_2, where c_2 contains constants only from c_1, the expert may estimate the probability that if $F_1(c_1)$ is true then $F_1(c_1) \wedge F_2(c_2)$ will be true as well. We will denote the former statistic by $SPr(F(x))$ and the latter by $\mathrm{SPr}(F_2(c_2)|F_1(c_1))$. For example, $\mathrm{SPr}\left(\mathrm{Cancer}(c)|\mathrm{Smokes}(c)\right) = 0.4$ means that if the chosen individual c smokes, (s)he has lung cancer as well with probability 0.4 according to the expert. Similarly, $\mathrm{SPr}\left(\mathrm{Smokes}(X)\right) = 0.01$ states the percentage of the population that smokes in the opinion of the expert. If the MLN happens to be *symmetric* in the sense that for any bindings of x_1 and x_2 to constant vectors c_1 and c_2, $\mathrm{SPr}(F_2(c_2)|F_1(c_1))$ is constant, we allow the notation $\mathrm{SPr}(F_2(x_2)|F_1(x_1))$ where x_2 only contains variables from x_1. *W.l.o.g.* we henceforth assume that $x_1 = x_2$ in any subjective probabilities.[1]

Let $g(F(x))$ denote the total number of groundings of formula $F(x)$ and let $\overline{\mu}_{F(x)} = \frac{\mu_{F(x)}}{g(F(x))}$. Given the definition of $\mathrm{SPr}(F_2(x)|F_1(x))$ and $\mathrm{SPr}(F(x))$, an initial idea would be to take $\overline{\mu}_{F(x)}$, $\overline{\mu}_{F_2(x)\wedge F_1(x)}$, and $\overline{\mu}_{F_1(x)}$, and try to satisfy $\overline{\mu}_{F(x)} = \mathrm{SPr}(F(x))$ and $\mathrm{SPr}(F_2(x)|F_1(x)) = \frac{\overline{\mu}_{F_2(x)\wedge F_1(x)}}{\overline{\mu}_{F_1(x))}}$ for every given subjective (conditional) probability, in absence of training data. In many cases, however, it is impossible to match the subjective probabilities of the expert. For example, consider the case where according to the expert $\mathrm{SPr}(P(x)) = 0.5$ and $\mathrm{SPr}(P(x) \vee Q(x)) = 0.4$. It is easy to see that in this situation no vector θ would provide a normalized μ that would match both subjective probabilities. We will call a set S of subjective (conditional) probabilities *inconsistent* in an MLN M that has all the formulas occuring in S if there does not exist any θ such that $\overline{\mu}_{F(x)}(\theta) = \mathrm{SPr}(F(x))$ and $\frac{\overline{\mu}_{F_2(x)\wedge F_1(x)}(\theta)}{\overline{\mu}_{F_1(x)}(\theta)} = \mathrm{SPr}(F_2(x)|F_1(x))$ for every $\mathrm{SPr}(F(x)), \mathrm{SPr}(F_2(x)|F_1(x)) \in S$. It can be proven that S is inconsistent in M if and only if there is not any distribution that would satisfy all the probabilistic constraints in S.

[1] Fisseler [4] explains in more details why conditional probability constraints must be dealt with at the ground level in Markov Logic-like relational probabilistic logics in order to match our definition for conditional probability. Thimm *et. al* [5] provide several different semantics for defining first-order conditional probabilities in probabilistic logics. The symmetric case described above corresponds to what they call *aggregating semantics*. It is beyond the scope of our paper to examine all the ways in which first-order conditional probabilities could be defined. For the sake of simplicity in rest of the paper we assume that subjective conditional probabilities are either defined at the ground level or are symmetric (*i.e.*, use aggregating semantics).

We will call a set of subjective (conditional) probabilities *fully specified* if for every subjective conditional probability $\mathrm{SPr}(F_2(x)|F_1(x))$, a value for $\mathrm{SPr}(F_1(x))$ is provided by the expert as well. In the case of fully specified subjective probabilities, we can replace a constaint involving $\mathrm{SPr}(F_2(x)|F_1(x))$ by the constraints $\overline{\mu}_{F_1(x)} = \mathrm{SPr}(F_1(x))$ and $\overline{\mu}_{F_2(x)\wedge F_1(x)} = \mathrm{SPr}(F_2(x) \wedge F_1(x))$.

In Sec. 5, we define a prior over θ assuming that the domain expert provides a consistent and fully specified set of subjective probabilities. This is a realistic assumption if the expert's subjective probabilities are not literally subjective, but have foundations in the statistics of real world data (*e.g.* 20% of US adults smoke). In Sec. 6, we allow inconsistent subjective conditional and non-conditional probabilities, with the tradeoff of possibly requiring greater computational effort.

4 Gaussian Priors and Chordal Graphs

Before describing our proposed solution for incorporating an expert's knowledge into an MLN, we discuss the idea of using non-zero mean Gaussian (or Laplace) priors to represent preference for subjective probability values [6]. We will demonstrate that using log-odds or log-probabilities as means of Gaussian (or Laplace) priors can be used under special circumstances. However, the standard deviation of each Gaussian needs to be scaled based on the associated probability of the formula, and rewriting an arbitrary MLN to put it into the required form may cause an exponential increase in size. Our examples require only the propositional subset of Markov Logic.

The Alchemy Tutorial [2] describes how one can convert a Bayesian Network into a propositional Markov Logic knowledge base. In the conversion, each entry in the conditional or non-conditional probability table for a node generates a clause whose weight is the negative log of the probability. In the problem at hand, however, we *begin* with an MLN, not a Bayesian Network.

Chordal graphs are the subset of undirected graphical models which correspond to both directed and undirected graphical models. We show that the problem of representing consistent expert knowledge in an MLN whose underlying Markov Random Field is chordal can be solved efficiently. Suppose we have a propositional knowledge base to which the corresponding ground MRF is a chordal graph G. It follows that the probability model P (represented by the ground Markov Network) is decomposable [7,8], *i.e.*, the joint probability of its random variables can be represented as the product of the joint probabilities of the variables in the individual cliques divided by the product of the joint probabilities of random variables in certain intersections of certain pairs of cliques. More precisely, let $C_1, \ldots, C_n$ be the sets of variables belonging to each maximal clique ordered by a maximum cardinality ordering, and let $C_{j(i)}$ be the unique predecessor of C_i in a join tree corresponding to G. Then the joint probability can be expressed as, $\Pr(\cup C_i = x) = \frac{\prod_i \Pr(C_i = c_i)}{\prod_i \Pr(C_i \cap C_{j(i)} = c_i \cap c_{j(i)})}$. For a clique C with variables $X_1, \ldots, X_n$ we will call a set S_C of conjunctions a *cover* of C if S_c contains all the possible 2^n different conjunctions over $X_1, \ldots, X_n$. To be able to

use the log probabilities, we require that the formulas present in the knowledge base contain exactly the conjunctions that cover every C_i clique and $C_i \cap C_{j(i)}$ intersection of cliques.

Assume now that there is a domain expert who specifies consistent probabilities for all the formulas in the knowledge base. Let T be a truth assignment to all the variables in the MLN. Let $t_{C=T}$ be a conjunction corresponding to the truth assignment in clique C (or intersection of cliques) that agrees with T, and let $p_{t_{C=T}}$ denote its probability (this probability can be unknown, *i.e.*, needed to be learned or specified by the expert). In this setting, the probability of a truth assignment T to all the variables can be written as:

$$\begin{aligned}\Pr(T) &= \frac{\prod_i p_{t_{C_i=T}}}{\prod_i p_{t_{C_i \cap C_{j(i)}=T}}} = \exp\Big(\sum_i \ln p_{t_{C_i=T}} - \sum_i \ln p_{t_{C_i \cap C_{j(i)}=T}}\Big) \qquad (5)\\ &= \exp\Bigg(\sum_i \sum_{T' \in \mathcal{T}_{C_i}} \ln p_{t_{C_i=T'}} f_{t_{C=T'}}(T)\\ &\quad - \sum_i \sum_{T' \in \mathcal{T}_{C_i \cap C_{j(i)}}} \ln p_{t_{C_i \cap C_{j(i)}=T'}} f_{t_{C_i \cap C_{j(i)}=T'}}(T)\Bigg),\end{aligned}$$

where $\mathcal{T}_C$ is the set of all truth assignments over the variables in clique C and $f_{t_{C=T'}}$ corresponds to the feature which represents the conjunction belonging to $t_{C=T'}$, *i.e.*, 1 if the conjunction is true, otherwise it is false. It is easy to see that the last line in (5) corresponds to a Markov Logic representation and that for every truth assignment T the MLN gives back the correct probability, with no normalization needed. Thus, chordal MLNs have the advantage that one can use log-probabilities as priors on the weights. However, MLNs are not, in general, chordal, and modifying an MLN to make it chordal — *i.e.* adding formulas that triangulate the underlying graph — can increase its size exponentially [9].

A second disadvantage of the approach just described is that Gaussian (or Laplace) priors on the natural parameters do not translate to Gaussian (or Laplace) priors in the mean parameter space — that is, in the space of probabilities as opposed to weights. Intuitively, one would want to use the variance of the prior to control how close the parameter is to the subjective probability after training. However, distance in the θ space does not linearly transform to a distance in the μ space. *E.g*, consider having one formula $F(x)$. A change of its weight from 0 to 1 would change its probability from 0.5 to ≈ 0.73, while a change from 1000 to 1001 would practically not change its probability. At a minimum, we would need to scale the standard deviations of the Gaussian priors according to the mean parameter of the distribution. This distinction is not present in (3).

Moreover, in many cases the expert can only know the (subjective) probabilities of a subset of formulas. Even if this subset of the formulas spans a chordal MLN satisfying the conditions to use log-probabilities, we still cannot use log-probabilities as weights if with the rest of the formulas altogether we do not have a chordal MLN. To illustrate this problem, consider the case

when we have $N+1$ ground atoms $p, q_1, \ldots, q_N$ in the domain and the expert knows exactly the probability of p being true, or equivalently provides the value of $o = \frac{\mathrm{SPr}(p)}{1-\mathrm{SPr}(p)}$. Further assume we also have formulas $p \vee q_i$ in our knowledge base for every $i = 1, .., N$, and learn the weights of these formulas from a training data set for which we know $\mu_p = \mathrm{SPr}(p)$ holds. If we use a strong non-zero mean Gaussian prior over the weight of p centered around $\log o$, *i.e.*, we fix the weight of p during the weight learning, then if w_i is the weight needed for $p \vee q_i$ to match the empirical feature count from the data, we will get $\frac{\mathrm{Pr}(p)}{1-\mathrm{Pr}(p)} = \frac{\mathrm{SPr}(p) \sum_{q_1,\ldots,q_N} \prod_{i=1}^{N} \exp(w_i)}{(1-\mathrm{SPr}(p)) \sum_{q_1,\ldots,q_N} \prod_{i=1}^{N} I[q_i] \exp(w_i)} = o \prod_{i=1}^{N} \frac{2\exp(w_i)}{\exp(w_i)+1}$, where $I[q_i]$ is the feature corresponding to $p \vee q_i$ with the substitution $p =$ false, *i.e.*, $I[q_i] = 1$ if q_i is *true*, otherwise 0. (We do not have a feature in the numerator, since $p \vee q_i$ is always true when $p =$ true.) It is easy to see that as we increase N our wrong choice of prior can cause an arbitrarily large deviation in the learned marginal of p from the correct one (consider the case when μ_p is close to 0, so using the expert's probability we set *e.g.*, $w_p = -100$ while $\mu_{p \vee q_i}$ is close to 1, and *e.g.* after the weight learning we have $w_i = 200$ for every i). We will demonstrate this downside of the prior further in Sec. 7 with experiments.

5 Defining a Prior on the Natural Parameters

In this section we consider the case when the expert gives us a fully specified set of consistent subjective probabilities, and show that we can avoid the disadvantages of Gaussian priors defined on the natural parameters. Exponential families of probability distributions have the advantageous property of easily defined conjugate priors [10,11] —

$$\Pr(\theta|\beta,\alpha) \propto \exp\left(\langle\theta, \alpha\beta\rangle - \alpha A(\theta)\right) \tag{6}$$

— and these priors have an intuitive meaning. β can be related to $\overline{f}(\mathcal{D})$ and α to N if we compare (6) to (3). β takes the role of the average empirical feature count, while α is going to be the pseudo-count. However, if β is not in the set of consistent mean parameters, then (6) is not going to form a distribution (*i.e.*, it is an *improper* prior).

Let $\theta_{F(x)}, \beta_{F(x)}$, and $\alpha_{F(x)}$ denote the components of the vectors which correspond to the feature $f_{F(x)}$. We can capture consistent subjective probabilities by setting $\beta_{F(x)} = \mathrm{SPr}(F(x))g(F(x))$. This intuitively makes sense, because the knowledge of the expert will be represented as pseudo-data, where α describes the confidence of the expert in his subjective probabilities, and β represent the sufficient statistic belonging to the pseudo-data. The posterior becomes:

$$\Pr(\theta|\mathcal{D},\beta,\alpha) \propto \exp\left(\left\langle\theta, \left(\alpha\beta + N\overline{f}(\mathcal{D})\right)\right\rangle - (\alpha+N)\, A(\theta)\right) . \tag{7}$$

The gradient of the logarithm of the posterior *w.r.t.* θ becomes:

$$\frac{\partial \log \Pr(\theta|\mathcal{D},\beta,\alpha)}{\partial\theta} = \alpha\beta + N\overline{f}(\mathcal{D}) - (\alpha+N)\mathbb{E}_\theta[f] . \tag{8}$$

This shows that the probability of the data and subjective probabilities of the expert are measured on the same scale; thus, we do not have to make adjustments for the subjective probabilities depending on their values, in contrast to the case for Gaussian priors.

The posterior is a log-concave function, which can be verified by taking the derivative of (8) *w.r.t.* θ. Thus, we are guaranteed to find a global maximum of (8) by using a gradient ascent.

This formulation assumes that the subjective probabilities of the expert and the data are coming from domains with the same size, which is not a realistic assumption in statistical relational learning. For example, a doctor can base his estimates on thousands of previously seen patients, while the training data can contain only the data for a small control group. We can allow the domain size to vary by using a distinct A for the data and for the expert in (6), (7), and (8) which would result in the computation of $\mathbb{E}[f]$ w.r.t. two different domain size of MLN but with the same knowledge base. This approach can be generalized to allow the different training examples to come from different sized domains, and to incorporate the knowledge of different experts. Although these modifications are straightforward, the effect of weak transfer [12] may change the meaning of subjective probabilities in different domains. For the sake of simplicity in the rest of the paper we will only consider the base case, *i.e.*, where the training data and the expert's knowledge come from the same domains.

In summary: the approach just described allows us to incorporate consistent prior knowledge by defining a prior over the natural parameters, avoids the problems of Gaussian priors (*i.e.* requiring a chordal structure and difficulty in defining the variance), and requires (only) solving a convex optimization problem. However, inconsistent subjective probabilities are difficult to handle in this approach. Furthermore, the expert is required to specify a subjective probability for every formula. A possible approach that could solve both issues would be to try to define a distribution (a hyper-prior) over the parameters (hyper-parameters) of the prior distribution. However, then for any reasonable choice of hyper-prior we would have the problem of dealing with a normalization factor in (6) that depends on the value of the hyper-parameters. Instead of going this route, in the next section we will define a prior over the mean parameters.

6 Defining Prior on the Mean Parameters

As we pointed out in the last section, defining a prior on θ only works when the subjective probabilities are consistent and the conditional probabilities are fully specified. To overcome these limitations, we soften the constraints by introducing a prior on the set of consistent μ values. Let $\Pi(\mu)$ be the prior on μ. Let $\Pr(\mathcal{D}|\theta) = \prod_{i=1}^{N} \Pr(x_i|\theta)$ be the probability of the i.i.d. training data given θ or μ. The posterior $\Pr(\mathcal{D}|\mu)$ is proportional to $\Pr(\mathcal{D}, \mu)$, hence it is sufficient to maximize $\Pr(\mathcal{D}, \mu)$. The log-probability of the data with the prior can be written as:

$$L(\mathcal{D}, \mu(\theta)) = \ln \Pr(\mathcal{D}|\mu) + \ln \Pi(\mu) \; . \tag{9}$$

We are looking for the weight vector (θ) that maximizes $L(\mathcal{D}, \mu(\theta))$. The gradient of L *w.r.t.* θ is:

$$\frac{\partial L}{\partial \theta} = \frac{\partial \ln \Pr(\mathcal{D}|\theta)}{\partial \theta} + \frac{\partial \ln \Pi(\mu)}{\partial \mu}\frac{\partial \mu}{\partial \theta} = \sum_i (f(d_i) - \mu) + \Sigma_\theta \frac{\partial \ln \Pi(\mu)}{\partial \mu}\,, \quad (10)$$

where we use the fact that $\frac{\partial \mu(\theta)}{\partial \theta} = \Sigma_\theta$, the covariance matrix of f *w.r.t.* the distribution defined by θ. The concaveness of (9) depends on the choice of $\ln \Pi(\mu)$; in general, $\ln \Pi(\mu)$ may be non-concave. Nonetheless, with a careful choice of $\Pi(\mu)$ there are cases when finding a global optimum of L can be reduced to a convex-optimization problem, or at least a solution with quality guarantees can be found by a gradient ascent algorithm.

6.1 Choosing the Prior

Our goal is that when no training data is available, we would like $\overline{\mu}_{F(x)}$ to be as close to $\mathrm{SPr}(F(x))$ as possible. Similarly, for conditional probabilities we want to match $\frac{\overline{\mu}_{F_2(x)\wedge F_1(x)}}{\overline{\mu}_{F_1(x))}}$ to $\mathrm{SPr}(F_2(x)|F_1(x))$. A prior that can capture this goal is, *e.g.*, a truncated Gaussian distribution over $\overline{\mu}_{F(x)}$, $\frac{\overline{\mu}_{F_2(x)\wedge F_1(x)}}{\overline{\mu}_{F_1(x))}}$ centered around $\mathrm{SPr}(F(x))$ and $\mathrm{SPr}(F_2(x)|F_1(x))$, respectively. We have to truncate the Gaussian since $\overline{\mu}$ is constrained to be between 0 and 1; moreover, the distribution has to be defined over consistent $\overline{\mu}$ values. Since $\overline{\mu}_{F_1(x))}$ can get close to 0, for numerical stability it is more beneficial to try to match $\overline{\mu}_{F_2(x)\wedge F_1(x)}$ to $\overline{\mu}_{F_1(x)}\mathrm{SPr}(F_2(x)|F_1(x))$. The distribution over μ is just a linear transformation of the truncated Gaussian distribution over $\overline{\mu}$ (since the Jacobian $\frac{\partial \overline{\mu}}{\partial \mu}$ is constant), resulting in:

$$\Pr(\mu) \propto \exp\left(-\sum_{F(x)} \alpha_{F(x)} \left(\overline{\mu}_{F(x)} - \mathrm{SPr}\left(F(x)\right)\right)^2 - \right. \quad (11)$$
$$\left. \sum_{F_2(x)|F_1(x)} \alpha_{F_2(x)|F_1(x)} \left(\overline{\mu}_{F_2(x)\wedge F_1(x)} - \overline{\mu}_{F_1(x)}\mathrm{SPr}\left(F_2(x)|F_1(x)\right)\right)^2\right)$$

In (11), the different α values correspond to the confidence of the expert in his different subjective probabilities, *e.g.* $\alpha = 0$ tells that the expert has no information about the subjective (conditional) probability of that feature.

6.2 Cases Solvable by Gradient Ascent

As we mentioned before, there are special cases when we end up with a convex optimization problem or can give guarantees about the quality of the solution of a gradient ascent algorithm. An example for the former is when there is no training data available, the subjective probabilities are consistent, and there are no subjective conditional probabilities given, since then $\Pi(\mu)$ takes its maximum when the exponent in (11) is 0. Consequently, we can just find the point where

$\forall F(x) : \overline{\mu}_{F(x)} - \mathrm{SPr}(F(x)) = 0$ using regular gradient ascent (without the presence of Σ_θ in (10)). (This is exactly equivalent to using the prior over θ defined in (6) where the same $\alpha_{F(x)}$ is used for every formula.)

Another important case is when we do not have a convex-optimization problem, but can still guarantee that all the points where the gradient is 0 are located close to a global optimum, uses a Gaussian prior for $\Pi(\mu)$. More precisely:

Proposition 1. *The gradient ascent algorithm always converges to a stationary point θ, where $\overline{\mu}(\theta)$ is guaranteed to fall within an $\epsilon(N)$ radius of $\hat{f}(\mathcal{D})$ if $\forall i$: $0 < \hat{f}_i(\mathcal{D}) < 1$, where $\hat{f}_{F(x)}(\mathcal{D}) = \frac{\overline{f}_{F(x)}(\mathcal{D})}{g(F(x))}$, N is the amount of training data and $\epsilon(N)$ is a strictly monotonically decreasing function of N and $\lim_{N\to\infty} \epsilon(N) = 0$.*

Proof. (Sketch) We start with showing that for all the θ's where this gradient is 0, it is true that $\overline{\mu}(\theta)$ has to be within $\epsilon(N)$ radius of $\hat{f}(\mathcal{D})$. Σ_θ is a covariance matrix of bounded valued random variables, hence the entries in Σ_θ are bounded as well. Also, both $\mathrm{SPr}(F(x)) - \overline{\mu}_{F(x)}$ and $\overline{\mu}_{F_2(x)\wedge F_1(x)} - \overline{\mu}_{F_1(x)}\mathrm{SPr}(F_2(x)|F_1(x)$ are bounded; consequently, the vector $g(\theta) = \Sigma_\theta \frac{\partial \ln \Pi(\mu)}{\partial \mu}$ is bounded, and there is a bound that does not depend on N or θ. Let the components of vector b be the tightest bounds on the absolute values of the components of g, *i.e.*, $b_i = \sup_\theta\{|g_i(\theta)|\}$. Let $g'(\theta) = f(\mathcal{D}) - N\mu(\theta) = N(\overline{f}(\mathcal{D}) - \mu(\theta))$. In (10) the gradient can only be 0 if $g' + g = 0$. A necessary condition for this $|g'_i(\theta)| \leq b_i$ for every $1 \leq i \leq n$, equivalently $|\overline{f}_i(\mathcal{D}) - \mu_i(\theta)| \leq \frac{b_i}{N}$, *i.e.*, $|\hat{f}_i(\mathcal{D}) - \overline{\mu}_i(\theta)| \leq \frac{b_i}{g_i(x)N} = \epsilon(N)$. Furthermore, the normalized gradient is directed towards the sphere centered around $\hat{f}(\mathcal{D})$ with radius $\epsilon(N)$. This completes the proof.

Since $\overline{f}(\mathcal{D})$ is the value to which μ would converge in absence of subjective probabilities, this result ensures that with sufficient training data μ converges to the same value it would converge to without having the prior, which is a desideratum for any prior. This results always holds as long as the gradient of $\ln \Pi(\mu)$ is bounded. The analogue of Proposition 1 holds, when we increase the domain size, instead of the number of training data sets. (*I.e.*, if the expert bases his statistics on seeing a population of 1000 people, the weight of his knowledge becomes negligible when we have a training data set with, *e.g.*, 1000000 people.)

Thus, despite that L can have more than one local optimum, a gradient ascent algorithm will serve as a basis for our optimization algorithm. We will empirically demonstrate that even when the conditions of the (approximation) guarantee do not hold, we can still be better off incorporating subjective probabilities rather than solely relying on the available data.

7 Experiments

Our implementation is an extension to the Probabilistic Consistency Engine (PCE) [13], an open source MLN inference tool that has been used effectively in different problem domains [13,14]. We used stochastic gradient ascent with a monotonically decreasing learning rate to find a point where the gradient is 0,

and added a regularizer so that in the guaranteed convex cases we would always end up with a unique solution. We also experimented with taking random steps occasionally to avoid becoming stuck in a local optimum and to increase our chance of reaching the global optimum. To compare the quality of the weight vectors that our weight learning algorithm visited, we would need to evaluate the log-posteriors at the visited points. However, since computing the real log-posteriors in (9) is costly, we use the combination of the pseudo log-likelihood of the data along with the log-probability of the prior for the comparison, to select the best weights. In our experiments, we always ended up using weight vectors where the gradient ascent algorithm with decreasing learning rate stopped, thus, simple gradient ascent provided satisfactory results in our examples. Note that, this can be due to the fact that some random noise is always present in the sampling algorithm, depending on how many samples we are using to approximate μ in (10) which helped get out from the local optima.

The goal of our experiments is to demonstrate the benefits of our model and show the advantages of using our proposed prior compared to the one currently available in Alchemy, namely Gaussian non-zero mean priors. In our first batch of experiments, we used an MLN that models the failure dependencies in a cyber-physical system, the Cabin Air Compressor (CAC) [15]. The CAC is an electromechanical subsystem in an Aircraft Environmental Control Systems (ECS), which is used to deliver fresh air to the cabin and provide pressurization. The MLN models a voting-3-CAC architecture, where three CAC subsystems are connected in a voting configuration, such that the overall system fails only if at least two out of the three CAC subsystems fail. A high load is put on a CAC if one or more of the other two CACs fail, and putting high load on a CAC increases its probability of failure. The MLN models the failure probability of the overall system, given the failure probabilities of each subsystem, and taking into account the effects of high load interactions. The voting-3-CAC MLN we used in our experiment had 4 hard and 4 soft clauses. We abbreviate predicates $failCac, failSystem, failCacHighLoad$ with C, S and H respectively. The hard and soft clauses we had in our KB are in Table 1. We resorted to the use of synthetic data, because in the real world system the probability of system failure is so low that acquiring real world data set that could capture the underlying distribution would require a lot of time. We hand-tuned the weights for the soft clauses to get a realistic model. We generated 10

Table 1. Hard clauses and soft clauses with their weights

Hard Clauses	
$\forall c,d,e,s : C(c) \wedge C(d) \wedge C(e) \wedge (c \neq d) \wedge (d \neq e) \wedge (c \neq e) \supset S(s)$	
$\forall c,d,e,s : C(c) \wedge C(d) \wedge \neg C(e) \wedge (c \neq d) \wedge (d \neq e) \wedge (c \neq e) \supset S(s)$	
$\forall c,d,e,s : C(c) \wedge \neg C(d) \wedge \neg C(e) \wedge H(d) \wedge H(e) \wedge (c \neq d) \wedge (d \neq e) \wedge (c \neq e) \supset S(s)$	
$\forall c,d,e,s : C(c) \wedge \neg C(d) \wedge \neg C(e) \wedge H(d) \wedge \neg H(e) \wedge (c \neq d) \wedge (d \neq e) \wedge (c \neq e) \supset S(s)$	
Soft Clauses	
-1.03594	$\forall c,d : failCac(c) \wedge \neg failCac(d) \wedge c \neq$d
-0.9857	$\forall c,d : failCacHighload(d) \wedge failCac(c) \wedge \neg failCac(d) \wedge c \neq d$
-0.491191	$\forall c : failCac(c)$
-1.01143	$\forall s : failSystem(s)$

synthetic training data sets each with 10, 20, 50, 100, 200, 500, 1000 and 10000 samples by collecting samples from the MLN using MC-SAT [16], and then selected only a subset of the samples in order to increase the independence between them. The normalized expected feature counts $\overline{\mu}$ varied between 0.01 to 0.3 for the soft clauses. We added a random value from $\{-\delta\overline{\mu}, +\delta\overline{\mu}\}$ to every normalized feature count to represent the uncertainty of the expert in his beliefs. We varied $\delta = 0.0, 0.1, 0.2, 0.3$ in our 4 different noise models (noiseless, noisy 1-3). In our experiments we set 3 subjective probabilities and 1 subjective conditional probability according to these sampled values (cases $noisy1$,$noisy2$ and $noisy3$ in Figure 1). Also, we used subjective probabilities computed from $\overline{\mu}$ for the noiseless case. We computed the KL-divergences $D_{KL}(P\|Q)$ between every learned distribution Q and the real distribution P, and averaged them for the 10 different sample sets. In the noiseless case, we used the log-posterior with the gradient specified in (8) (fully specified set of subjective probabilities). In the noisy cases, we used gradient ascent according to the gradient in (10) using the prior in (11). We set the weight of our prior to match approximately 100 samples from the training data. As we see from the figure, the KL-divergence is relatively high without using our prior for 10 samples, and around 500 samples starts converging to the same values in all cases. This can be explained by the fact that for certain formulas, $\overline{\mu}$ has low values – so consequently for a few samples the formulas corresponding to the low expected average feature counts are unlikely to be satisfied in any of the generated samples. The KL-divergence is finite in these cases due to the use of a regularizer, which prevents the weights from becoming infinite. Arguably other measures (*e.g.* L_1-norm) could be used instead of KL-divergence. However, in case of a fault tolerant system, one has to consider the associated penalty when a system is claimed to be fail-safe, but in reality has a non-zero probability for failure – L_1 distance measure would not capture this.

In summary, when small amount of training data is used, it is beneficial to train the MLN using the subjective probabilities of a domain expert, even if he is not completely confident in his subjective probabilities.

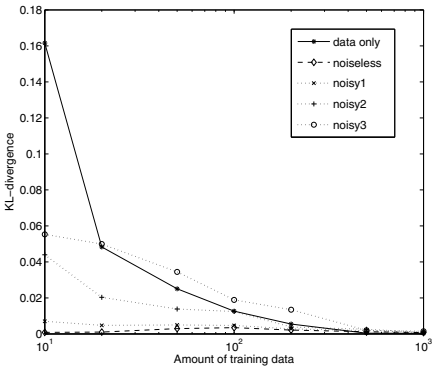


Fig. 1. The averaged KL-divergences measured from the true distribution for the 4 noise models

Table 2. Knowledge bases for the different experiments

Formula No.	Formula
1	$\forall x : Smokes(x)$
2	$\forall x : Smokes(x) \wedge Cancer(x)$
3	$\forall x : Smokes(x) \wedge \neg Cancer(x)$
4	$\forall x : \neg Smokes(x) \wedge Cancer(x)$
5	$\forall x : \neg Smokes(x) \wedge \neg Cancer(x)$
6	$\forall x, y : Friends(x, y) \wedge Smokes(x) \supset Smokes(y)$
7	$\forall x, y : Relatives(x, y) \wedge Cancer(x) \supset Cancer(y)$

In our second set of experiments, we used a modified version of the "smoking" social network example MLN from the Alchemy Tutorial to analyze the problems of using log-probabilities as weights. We created two different knowledge bases A and B for our experiments. Both knowledge bases used the formulas from Table 2; knowledge base A used 1-5 while B from 1-7. For each knowledge base we ran two sets of experiments, one with using our prior in the mean parameter space, and one with using Gaussians in the natural parameter space centered around the logarithm of the value of the appropriate (conditional) probabilities. The goal of the experiments were to use a strong prior (with small variance and high α, respectively for the non-zero mean Gaussian over the weights and our truncated Gaussian prior,) forcing the formulas $1-5$ which only appear in KB A to have log-probability weight/subjective probability to be equal to the values provided by the expert. About the probabilities of formulas $6-7$ the expert had no information, hence their weights could vary freely in both cases during the weight learning. We had 8 people in the domain. We considered 3 sets of weights and generated 100 samples from the distribution represented by the MLN that had all the formulas in the table. We again created our training data sets by using samples from MC-SAT. We computed the feature counts from the samples and, after normalizing them, we set the (log) probabilities and conditional probabilities of the appropriate formulas in Table 2. In the experiments which used KB A, both priors performed similarly. However, using log-probability weights for formulas 1-5 in KB B proved to be a bad estimate – this is because after weight learning, the formulas 6-7 had non-zero weights, thereby changing the probabilities of formulas 1-5 whose weights were fixed using the desired values. Because the domain was too large to compute the KL-divergence, we measured the L_1-distance between the normalized value of the expected feature counts (probabilities) we get for formulas 1-5, while using the same priors in KB A and B. The normalized feature counts of formulas 1-7 in the 3 experiments and the measured L_1-distances are in Tables 3 and 4. These experiments confirmed that using log-probability weights as means of Gaussian priors can decrease the quality of the learned distribution, even if the expert has access to the true probability values for a subset of the formulas, and motivates the use of our proposed prior.

Table 3. The probabilities of the formulas in the different experiments

Exp.no.	Probabilities of Formulas						
	1	2	3	4	5	6	7
1	0.6058375	0.4609375	0.144825	0.00741250000002	0.38695	0.803438	0.973437
2	0.7480875	0.711075	0.0367375	0.0083125	0.2441	0.862812	0.963750
3	0.68445	0.6442875	0.0404375	0.0749499999999	0.2404625	0.839688	0.943125

Table 4. The L_1 distances between the normalized expected feature counts and the empirical feature counts, using the two different priors

Experiment	L_1 distance	
	our proposed prior	log-probability weights
1	< 0.05	0.524025
2	< 0.05	0.2613875
3	< 0.05	0.2862375

8 Related Work

How the knowledge of an expert can be represented as parameter constraints for Bayesian Networks is discussed in [17,18]. Although their formalisms can incorporate complex knowledge, it is still restricted to Bayesian networks. Prior probability distributions within the framework of Bayesian statistics in Markov Random Fields have been used successfully in many domains, such as computer vision [19,20]. For exponential families other methods have been proposed to incorporate information based on expectations of features *e.g.* in [21] or *measurements* in [22], but their models are not tailored to MLNs and *e.g.*, do not allow directly constraining the ratio of expectations of features which is needed to capture conditional probabilities of formulas. The closest relevant research on representing knowledge given as (conditional) probabilities of formulas in MLNs is the work described in [4]. There, the language of Markov logic is extended by probabilistic constraints, which can either be probabilities of universally quantified formulas or conditional probabilities of propositional formulas. However, their framework only handles consistent constraints and does not allow combining the prior knowledge with training data. Eliciting probabilities from expert opinion has been used in research in other areas. For example, [23] shows how to extend de Finettis betting-odds method for considering subjective beliefs regarding ambiguous events.

9 Conclusion

In this paper, we presented a mathematical framework for incorporating subjective probabilities into a Markov Logic Network within the framework of Bayesian statistics. We discussed the benefits and limitations of defining the prior over the natural parameters (weights) versus the mean parameters (probabilities) of the MLN, and demonstrated that earlier approaches based on Gaussian priors over the natural parameters were inadequate. Our framework allows knowledge about

conditional subjective probabilities to be incorporated as well as non-conditional probabilities, and can extended to priors where subjective probabilities come from multiple experts. When we are provided with a fully specified and consistent set of subjective probabilities, defining the prior on the natural parameters results in a convex-optimization problem, which is an advantage in terms of computational effort. It is often the case, however, that domain expert subjective probabilities are not consistent. We showed how to make use of possibly inconsistent subjective probabilities by defining a prior over the mean parameters. This approach allows for more flexibility, but at possibly greater computational cost for learning, because the optimization problem may be non-convex. However, we provided conditions under which the optimization problem may still be well-approximated by simple gradient ascent. In future work, we plan on investigating specific real-world applications where it is important to combine expert knowledge with data, such as medical expert systems, and where the expert can give constraints different from what we discussed in this paper. Although, we only defined our priors in the context of Markov logic, most of our results can be generalized for exponential families in a straightforward way.

Acknowledgments. This material is based upon work supported by the DARPA Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181, and by ARO grant W911NF-08-1-0242, ONR grant N00014-11-10417, Intel STCPC and NSF grant IIS-1012017. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, the Air Force Research Laboratory (AFRL), ARO, ONR, Intel, NSF or the US government. We would like to thank Hung Bui, Tuyen Ngoc Huynh for their helpful discussions during the problem formulation, Natarajan Shankar, Sam Owre for their help with PCE and valuable feedback, and Patrick Lincoln, David Israel for their support and insights.

References

1. Domingos, P., Lowd, D.: Markov Logic: An Interface Layer for Artificial Intelligence. In: Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers (2009)
2. Kok, S., Sumner, M., Richardson, M., Singla, P., Poon, H., Lowd, D., Wang, J., Nath, A., Domingos, P.: The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington (2010)
3. Geiger, D., Meek, C.: Graphical models and exponential families. In: Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 156–165. Morgan Kaufmann, Madison (August 1998)
4. Fisseler, J.: Toward markov logic with conditional probabilities. In: FLAIRS Conference, pp. 643–648 (2008)
5. Thimm, M., Kern-Isberner, G., Fisseler, J.: Relational Probabilistic Conditional Reasoning at Maximum Entropy. In: Liu, W. (ed.) ECSQARU 2011. LNCS, vol. 6717, pp. 447–458. Springer, Heidelberg (2011)

6. Poon, H., Domingos, P.: Joint unsupervised coreference resolution with markov logic. In: EMNLP, ACL, pp. 650–659 (2008)
7. Pearl, J.: Probabilistic reasoning in intelligent systems - networks of plausible inference. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann (1989)
8. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
9. Chung, F.R.K., Mumford, D.: Chordal completions of planar graphs. J. Comb. Theory, Ser. B 62(1), 96–106 (1994)
10. Raiffa, H., Schlaifer, R.: Applied statistical decision theory [by] Howard Raiffa and Robert Schlaifer. In: Division of Research, Division of Research, Graduate School of Business Adminitration, Harvard University, Boston (1961)
11. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. Springer (2007)
12. Jain, D., Barthels, A., Beetz, M.: Adaptive Markov Logic Networks: Learning Statistical Relational Models with Dynamic Parameters. In: 19th European Conference on Artificial Intelligence (ECAI), pp. 937–942 (2010)
13. Ghosh, S., Shankar, N., Owre, S.: Machine reading using markov logic networks for collective probabilistic inference. In: Proceedings of ECML-CoLISD 2011 (2011)
14. Ghosh, S., Shankar, N., Owre, S., David, S., Swan, G., Lincoln, P.: Markov logic networks in health informatics. In: Proceedings of ICML-MLGC 2011 (2011)
15. Denker, G., Briesemeister, L., Elenius, D., Ghosh, S., Mason, I., Tiwari, A., Bhatt, D., Hailu, H., Madl, G., Nikbin, S., Varadarajan, S., Bauer, G., Steiner, W., Koutsoukos, X., Levendovsky, T.: Probabilistic, compositional, multi-dimension model-based verification (promise)
16. Poon, H., Domingos, P.: Sound and efficient inference with probabilistic and deterministic dependencies. In: AAAI (2006)
17. Niculescu, R.S., Mitchell, T.M., Rao, R.B.: Bayesian network learning with parameter constraints. Journal of Machine Learning Research 7, 1357–1383 (2006)
18. Campos, C.P., Tong, Y., Ji, Q.: Constrained Maximum Likelihood Learning of Bayesian Networks for Facial Action Recognition. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 168–181. Springer, Heidelberg (2008)
19. Geman, S., Geman, D.: Readings in computer vision: issues, problems, principles, and paradigms, pp. 564–584. Morgan Kaufmann Publishers Inc., San Francisco (1987)
20. Li, S.Z.: A markov random field model for object matching under contextual constraints. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 866–869 (1994)
21. Druck, G., Mann, G., McCallum, A.: Learning from labeled features using generalized expectation criteria. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, pp. 595–602. ACM, New York (2008)
22. Liang, P., Jordan, M.I., Klein, D.: Learning from measurements in exponential families. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, pp. 641–648. ACM, New York (2009)
23. Diecidue, E., Wakker, P., Zeelenberg, M.: Eliciting decision weights by adapting de finetti's betting-odds method to prospect theory. Open Access publications from Tilburg University urn:nbn:nl:ui:12-225938, Tilburg University (2007)

Score-Based Bayesian Skill Learning

Shengbo Guo[1], Scott Sanner[2], Thore Graepel[3], and Wray Buntine[2]

[1] Xerox Research Centre Europe
[2] NICTA and the Australian National University
[3] Microsoft Research Cambridge

Abstract. We extend the Bayesian skill rating system of TrueSkill to accommodate score-based match outcomes. TrueSkill has proven to be a very effective algorithm for matchmaking — the process of pairing competitors based on similar skill-level — in competitive online gaming. However, for the case of two teams/players, TrueSkill only learns from win, lose, or draw outcomes and cannot use additional match outcome information such as scores. To address this deficiency, we propose novel Bayesian graphical models as extensions of TrueSkill that (1) model player's offence and defence skills separately and (2) model how these offence and defence skills interact to generate score-based match outcomes. We derive efficient (approximate) Bayesian inference methods for inferring latent skills in these new models and evaluate them on three real data sets including Halo 2 XBox Live matches. Empirical evaluations demonstrate that the new score-based models (a) provide more accurate win/loss probability estimates than TrueSkill when training data is limited, (b) provide competitive and often better win/loss classification performance than TrueSkill, and (c) provide reasonable score outcome predictions with an appropriate choice of likelihood — prediction for which TrueSkill was not designed, but which can be useful in many applications.

Keywords: variational inference, matchmaking, graphical models.

1 Introduction

In online gaming, it is important to pair players or teams of players so as to optimise their gaming experience. Game players often expect competitors with comparable skills for the most enjoyable experience; match experience can be compromised if one side consistently outperforms the other. *Matchmaking* attempts to pair players such that match results are close to being even or a draw. Hence, a prerequisite for good matchmaking is the ability to predict future match results correctly from historical match outcomes — a task that is often cast in terms of latent skill learning.

TrueSkill [5] is a state-of-the-art Bayesian skill learning system: it has been deployed in the Microsoft Xbox 360 online gaming system for both matchmaking and player ranking. For the case of two teams/players, TrueSkill, like Elo [4], is restricted to learn skills from match outcomes in terms of win, lose, or draw

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 106–121, 2012.

(WLD). While we conjecture that TrueSkill discards potentially valuable skill information carried by score-based outcomes, there are at least two arguments in favour of TrueSkill's WLD-based skill learning approach:

- WLD-based systems can be applied to any game whose outcome space is WLD, no matter what the underlying scoring system is.
- In many games, the objective is not to win by the highest score differential, but rather simply to win. In this case, it can be said that TrueSkill's skill modeling and learning from WLD outcomes aligns well with the players' underlying objective.

On the other hand, we note that discarding score results ignores two important sources of information:

- High (or low) score differentials can provide insight into relative team strengths.
- Two dimensional score outcomes (i.e., a score for each side) provide a direct basis for inferring separate offense and defense strengths for each team, hence permitting finer-grained modeling of performance against future opponents.

In this work, we augment the TrueSkill model of WLD skill learning to learn from score-based outcomes. We explore single skill models as well as separate offense/defense skill models made possible via score-based modeling. We also investigate both Gaussian and Poisson score likelihood models, deriving a novel variational update for approximate Bayesian inference in the latter case. We evaluate these novel Bayesian score-based skill-learning models in comparison to TrueSkill (for WLD outcomes) on three datasets: 14 years of match outcomes for the UK Premier League, 11 years of match outcomes for the Australian Football (Rugby) League (AFL), and three days covering 6,000+ online match outcomes in the Halo 2 XBox video game. Empirical evaluations demonstrate that the new score-based models (a) provide more accurate win/loss probability estimates than TrueSkill (in terms of information gain) with limited amounts of training data, (b) provide competitive and often better win/loss classification performance than TrueSkill (in terms of area under the curve), and (c) provide reasonably accurate score predictions with an appropriate likelihood — prediction for which TrueSkill was not designed but important in cases such as tournaments that rank (or break ties) by points, professional sports betting and bookmaking, and game-play strategy decisions that are dependent on final score projections.

2 Skill Learning Using TrueSkill

Since our score-based Bayesian skill learning contributions build on TrueSkill [5], we begin with a review of the TrueSkill Bayesian skill-learning graphical model for two single-player teams. We note that TrueSkill itself allows for matches involving more than two teams and learning team members' individual performances, but these extensions are not needed for the application domains considered in the paper.

Suppose there are n teams available for pairwise matches in a game. Let $M = \{i, j\}$ specify the two teams participating in a match and define the outcome $o \in \{\textit{team-i-win}, \textit{team-j-win}, \textit{draw}\}$. TrueSkill models the probability $p(o|\boldsymbol{l}, M)$ of o given the skill level vector $\boldsymbol{l} \in \mathbb{R}^n$ of the teams in M, and estimates posterior distributions of skill levels according to Bayes' rule

$$p(\boldsymbol{l}|o, M) \propto p(o|\boldsymbol{l}, M)p(\boldsymbol{l}), \tag{1}$$

where a factorising Gaussian prior is assumed:

$$p(\boldsymbol{l}) := \prod_{i=1}^{n} \mathcal{N}(l_i; \mu_i, \sigma_i^2). \tag{2}$$

To model the likelihood $p(o|\boldsymbol{l}, M)$, each team i is assumed to exhibit a stochastic performance variable $p_i \sim \mathcal{N}(p_i; l_i, \beta^2)$ in the game [1]. From this we can model the performance differential d as an indicator function $p(d|\boldsymbol{p}, M) = \delta(d = p_i - p_j)$ and finally the probability of each outcome o given this differential d:

$$p(o|d) = \begin{cases} o = \textit{team-i-win}: & \mathbb{I}[d > \epsilon] \\ o = \textit{team-j-win}: & \mathbb{I}[d < -\epsilon] \\ o = \textit{draw}: & \mathbb{I}[|d| \leq \epsilon], \end{cases} \tag{3}$$

where $\mathbb{I}[\cdot]$ is an indicator function. Then the likelihood $p(o|\boldsymbol{l}, M)$ in (1) can be written as

$$p(o|\boldsymbol{l}, M) = \int \cdots \int_{\mathbb{R}^n} \int_{-\infty}^{+\infty} p(o|d)p(d|\boldsymbol{p}, M) \prod_{i=1}^{n} p(p_i|l_i) \, \mathrm{d}\boldsymbol{p} \, \mathrm{d}d.$$

The entire TrueSkill model relevant to M is shown in the factor graph of Figure 1 with $P(o|d)$ given for the case of $o = \textit{team-i-win}$. TrueSkill uses message passing to infer the posterior distribution in (1) — note that the posterior over l_i and l_j will be updated according to the match outcome while the posterior over l_k ($k \notin \{i, j\}$) will remain unchanged from the prior. An optimal message passing schedule in the TrueSkill factor graph (Figure 1) is provided in the caption; the message along arrow 2 is a step function that leads to intractability for exact inference and thus TrueSkill uses message approximation via moment matching.

TrueSkill is an efficient and principled Bayesian skill learning system. However, due to its design goals, it discards score information and does not take into account associated domain knowledge such as offence/defence skill components. Next, we propose extensions of the TrueSkill factor graph and (approximate) inference algorithms for score-based Bayesian skill learning, which address these limitations.

[1] Note that we sometimes abuse notations on the use of p, p_i and $\boldsymbol{p}$. p is a probability measure; p_i and $\boldsymbol{p}$ represent performance variables. The meaning of them is clear from the context.

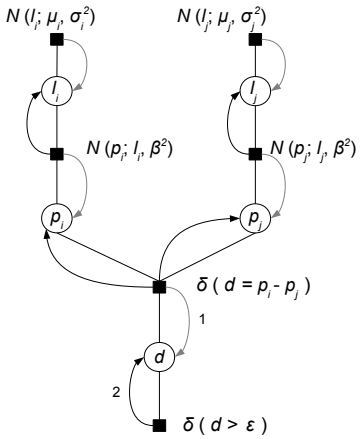


Fig. 1. TrueSkill factor graph for a match between two single-player teams with team i winning. There are three types of variables: l_i for the skills of all players, p_i for the performances of all players and d the performance difference. The first row of factors encode the (product) prior; the product of the remaining factors characterizes the likelihood for the game outcome team i winning team j. The arrows show the optimal message passing schedule: (1) messages pass along *gray* arrows from top to bottom, (2) the marginal over d is updated via message 1 followed by message 2 (which requires moment matching), (3) messages pass from bottom to top along *black* arrows.

3 Score-Based Bayesian Skill Models

In this section, we introduce three graphical models as extensions for the TrueSkill factor graph (Figure 1) to incorporate score-based outcomes in skill learning. Our first two graphical models are motivated by modeling score-based outcomes as generated by separate offence and defence skills for each team. The first generative score model uses a Poisson, which is natural model when scores are viewed as counts of scoring events. The second generative model uses a simpler Gaussian model. Our third model is a simplified version of the Gaussian model, which like TrueSkill, only models a single skill per team (not separate offence/defence skills) and places a Gaussian likelihood on the score difference, which may be positive or negative. Next we formulate each model in detail.

3.1 Offence and Defence Skill Models

In a match between two teams i and j producing respective scores $s_i \in \mathbb{Z}$ and $s_j \in \mathbb{Z}$ for each team, it is natural to think of s_i as resulting from i's offence skill $o_i \in \mathbb{R}$ and j's defence skill $d_j \in \mathbb{R}$ (as expressed in any given match) and likewise for j's score as a result of j's offence skill $o_j \in \mathbb{R}$ and i's defence skill $d_i \in \mathbb{R}$. This is contrasted with the univariate skill estimates of team i's skill l_i and team j's skill l_j used in TrueSkill, which lump together offence and defence skills for each team.

Given scores s_i and s_j for teams i and j, we model the generation of scores from skills using a conditional probability $p(s_i, s_j | o_i, o_j, d_i, d_j)$. We assume that

team i's score s_i depends only on o_i and d_j and likewise that team j's score s_j depends only on o_j and d_i:

$$p(s_i, s_j|o_i, o_j, d_i, d_j) = p(s_i|o_i, d_j)p(s_j|o_j, d_i). \tag{4}$$

Like TrueSkill, we assume that the joint marginal over skill priors independently factorises:

$$p(o_i, o_j, d_i, d_j) = p(o_i)p(d_j)p(o_j)p(d_i). \tag{5}$$

Given an observation of scores s_i for team i and s_j for team j, the problem is to update the posterior distributions over participating teams' offence and defence skills. According to Bayes rule and the previous assumptions, the posterior distribution over (o_i, o_j, d_i, d_j) is given by

$$\begin{aligned} p(o_i, d_i, o_j, d_j|s_i, s_j) &\propto p(s_i, s_j|o_i, d_i, o_j, d_j)p(o_i, d_i, o_j, d_j) \\ &\propto [p(s_i|o_i, d_j)p(o_i)p(d_j)]\ [p(s_j|o_j, d_i)p(o_j)p(d_i)]. \end{aligned} \tag{6}$$

Here we observe that estimating $p(o_i, d_i, o_j, d_j|s_i, s_j)$ factorises into the two independent inference problems:

$$p(o_i, d_j|s_i) \propto p(s_i|o_i, d_j)p(o_i)p(d_j), \text{and} \tag{7}$$

$$p(o_j, d_i|s_j) \propto p(s_j|o_j, d_i)p(o_j)p(d_i). \tag{8}$$

All models considered in this paper (including TrueSkill) assume Gaussian priors on team i's offence and defence skills, i.e., $p(o_i) := \mathcal{N}(o_i; \mu_{oi}, \sigma_{oi}^2)$ and $p(d_i) := \mathcal{N}(d_i; \mu_{di}, \sigma_{di}^2)$. Our objective then is to estimate the means and variances for the posterior distributions of $p(o_i, d_j|s_i)$ and $p(o_j, d_i|s_j)$. So far, the only missing pieces in this skill posterior update are the likelihoods $p(s_i|o_i, d_j)$ and $p(s_j|o_j, d_i)$ that specify how team i and j's offence and defence skills probabilistically generate observed scores. For this we discuss two possible models in the following subsections.

Poisson Offence/Defence Skill Model. Following TrueSkill, we model the generation of match outcomes (in our case, team scores) based on stochastic offence and defence *performances* that account for day-to-day performance fluctuations. Formally, we assume that team i exhibits offence performance $p_{oi} := \mathcal{N}(p_{oi}; o_i, \beta_o^2)$ and defence performance $p_{di} := \mathcal{N}(p_{di}; d_i, \beta_d^2)$. With these performances, we model team i's score s_i as generated from the following process: team i's offence performance p_{oi} promotes the scoring rate while the defence performance p_{dj} inhibits this scoring rate, the difference $p_{oi} - p_{dj}$ being the effective scoring rate of the offence against the defence.

Finally, we model the score by $s_i \sim \text{Poisson}(\lambda)$, where a requirement of a positive rate λ for the Poisson distribution requires the use of $\lambda = \exp(p_{oi} - p_{dj})$ since $p_{oi} - p_{dj}$ may be negative.[2] Likewise, one can model s_j by applying the same

[2] This exponentiation of $p_{oi} - p_{dj}$ may seem to be made only to ensure model correctness, but we show experimentally that it has the benefit of allowing the Poisson model to accurately predict scores in high-scoring games even when team skills are very close (and hence $p_{oi} - p_{dj} \approx 0$).

strategy when given $\lambda = \exp(p_{oj} - p_{di})$. We represent the resulting *Poisson-OD* model in Figure 2(P) where the joint posterior is

$$p(o_i, d_j, p_{oi}, p_{dj}|s_i) \propto p(s_i|p_{oi}, p_{dj})p(p_{oi}|o_i)p(p_{dj}|d_j)p(o_i)p(d_j),$$
$$p(o_j, d_i, p_{oj}, p_{di}|s_j) \propto p(s_j|p_{oj}, p_{di})p(p_{oj}|o_j)p(p_{di}|d_i)p(o_j)p(d_i).$$

We are only interested in the posterior distributions of o_i, d_j and o_j, d_i given s_i and s_j, respectively. Thus, we integrate out the latent performance variables to obtain the desired posteriors

$$p(o_i, d_j|s_i) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p(o_i, d_j, p_{oi}, p_{dj}|s_i)\mathrm{d}p_{oi}\mathrm{d}p_{dj},$$
$$p(o_j, d_i|s_j) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p(o_j, d_i, p_{oj}, p_{di}|s_j)\mathrm{d}p_{oj}\mathrm{d}p_{di}.$$

Like TrueSkill, we use Bayesian updating to update beliefs in the skill levels of both teams in a pairwise match based on the score outcome, thus leading to an online learning scheme. Posterior distributions are approximated to be Gaussian and used as the priors in order to learn each team's skill for the next match. Approximate belief updates via variational Bayesian inference in this model will be covered in Section 4.2.

Gaussian Offence/Defence Skill Model. An alternative to the previous Poisson model is to model $s_i \in \mathbb{R}$ and assume it is generated as $s_i \sim \mathcal{N}(\mu, \gamma^2)$, where $\mu = p_{oi} - p_{dj}$. One can similarly model s_j by applying the same strategy when given $\mu = p_{oj} - p_{di}$. We note that unlike the Poisson model, μ can be negative here so we need not exponentiate it. While this allows us to directly model match outcomes that allow negative team scores (c.f., Halo2 as discussed in Section 5.1), it is problematic for other match outcomes that only allow non-negative team scores. One workaround would be to introduce a truncated Gaussian model to avoid the problem of assigning non-zero probability to negative scores, but we avoid this complication in exchange for the simple and exact updates offered by a purely Gaussian model.

We show the resulting *Gaussian-OD* model in Figure 2(N), which differs from our proposed Poisson model only in modeling the observed score s_i (s_j) for team i (j) given the univariate performance difference variable x (y). In this model, all messages passed during inference are Gaussian, allowing for efficient and exact belief updates.

3.2 Gaussian Score Difference (SD) Model

Again assuming $s_i \in \mathbb{R}$ and $s_j \in \mathbb{R}$, algebra for the performance means in Figure 2(N) gives:

$$s_i = p_{oi} - p_{dj}, \qquad s_j = p_{oj} - p_{di}. \tag{9}$$

This implies

$$\begin{aligned} s_i - s_j &= (p_{oi} - p_{dj}) - (p_{oj} - p_{di}) \\ &= \underbrace{(p_{oi} + p_{di})}_{p_{li}} - \underbrace{(p_{oj} + p_{dj})}_{p_{lj}}, \end{aligned} \tag{10}$$

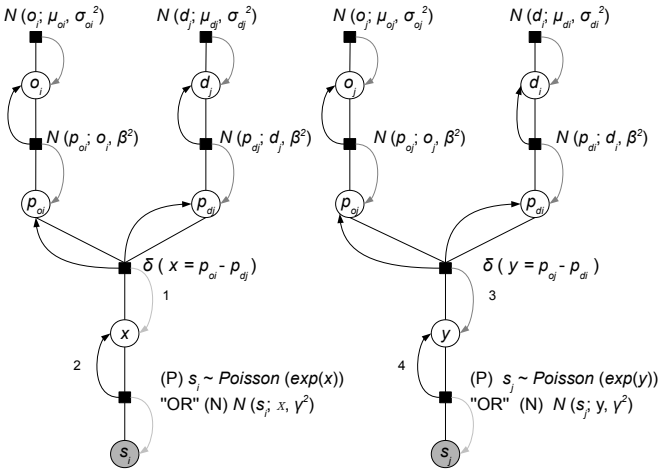


Fig. 2. The Poisson-OD (P) and Gaussian-OD (N) variants of TrueSkill factor graph for skill update of two teams based on the match score outcome (Left: modeling s_i; Right: modeling s_j). Note that the Poisson-OD and Gaussian-OD graphical models are merged due to limited space. Note also that the score observation factors use the Poisson distribution for the Poisson-OD model and the normal distribution for the Gaussian-OD model. The shaded variables are the observed ones. For each team i, it is characterized by offence skill o_i (the offence skill of team i) and defence skill d_i (the defence skill of team i). Given s_j for team j, the posterior distributions over (o_i, d_j) are inferred via message passing.

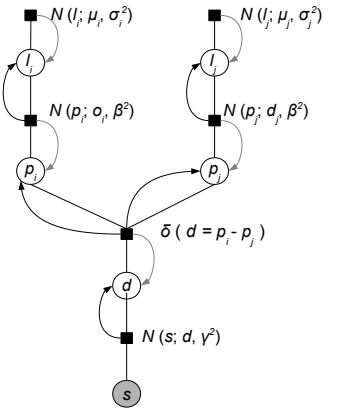


Fig. 3. Gaussian-SD model for skill learning from score differences. Both team i and team j are characterized by skill level l_i and l_j, respectively. The shaded variable s ($s = s_i - s_j$) denotes the score difference between s_i and s_j. Bayesian inference for the posterior skill level distributions has a closed-form solution.

which is like modeling the score difference with performance expressions p_{li} and p_{lj} of respective univariate skill levels, l_i and l_j. Motivated by (9), we propose a score difference (SD) Gaussian model that uses a likelihood model for the observed difference $s := s_i - s_j$ specified as $s \sim \mathcal{N}(p_{li} - p_{lj}, \gamma^2)$ as shown in Figure 3.

4 Skill and Win Probability Inference

We infer skill distributions in all proposed models via online Bayesian updating. While exact inference in the purely Gaussian models can be achieved by solving linear systems, Bayesian updating provides an efficient (also exact) incremental learning alternative. Equations for Bayesian updates and win probability inference are model-dependent and presented below.

4.1 Inference in TrueSkill

Bayesian Update: The Bayesian update equations in the TrueSkill model (Figure 1) are presented in [5].

Win Probability: Given skill levels of team i and j, $l_i \sim \mathcal{N}(l_i; \mu_i, \sigma_i^2)$ and $l_j \sim \mathcal{N}(l_j; \mu_j, \sigma_j^2)$, we first compute the distribution over performance difference variable d, and get $d \sim \mathcal{N}(d; \mu_d, \sigma_d^2)$ with $\mu_d = \mu_i - \mu_j$ and $\sigma_d^2 = \sigma_i^2 + \sigma_j^2 + 2\beta^2$. The winning probability of team i is given by the probability $p(d > 0)$ defined as

$$p(d > 0) = 1 - \Phi\left(\frac{-\mu_d}{\sigma_d}\right), \tag{11}$$

where $\Phi(\cdot)$ is the normal CDF.

4.2 Inference in Poisson-OD Model

Bayesian Update: Some of the update equations in the Poisson-OD model (Figure 2(P)) have been presented in [5], with the exception of the marginal distribution over x and the message passing from the Poisson factor to x. Given a prior Gaussian distribution over x, $\mathcal{N}(x; \mu, \sigma^2)$, we next demonstrate how to update the belief on x when observing team i's score s_i.

By the sum-product algorithm [7], the marginal distribution of x is given by a product of messages

$$p(x|s_i) = m_{\delta \to x}(x) m_{s_i \to x}(x). \tag{12}$$

To avoid cluttered notation, let us use $m_1(x)$ to represent $m_{\delta \to x}(x) = \mathcal{N}(x; \mu, \sigma^2)$, i.e., the message passing from the factor $\delta(\cdot)$ to x, and $m_2(x)$ for $m_{s_i \to x}(x) = Poisson(s_i; \exp(x))$, i.e., the message passing from the Poisson factor to x (c.f., messages labeled 1 and 2 in Figure 2(P)). Due to the multiplication of $m_1(x)$ and $m_2(x)$, the exact marginal distribution of $p(x|s_i)$ is not Gaussian, which makes exact inference intractable. To maintain a compact representation of offence and defence skills, one can approximate $p(x|s_i)$ within a variational Bayes framework

by choosing a Gaussian distribution $q(x)^* : \mathcal{N}(x; \mu_{\text{new}}, \sigma^2_{\text{new}})$ that minimizes the KL divergence between $p(x|s_i)$ and $q(x)$, i.e.,

$$q(x)^* = \arg\min_{q(x)} \text{KL}\left[q(x)||p(x|s_i)\right]. \tag{13}$$

We derive a fixed-point approach for optimizing $q(x)$ [12] and describe this approach below.

Minimizer $q(x)$ for KL$(q(x)||p(x|s_i))$: We first expand the KL-divergence into its definition:

$$\begin{aligned}\text{KL}\left(q(x)||p(x|s_i)\right) &= \int q(x) \log\left(\frac{q(x)}{p(x|s_i)}\right) dx \\ &= -\log\sqrt{2\pi e \sigma^2_{new}} - E_{x\sim q(x)} \log\left(p(x|s_i)\right), \end{aligned} \tag{14}$$

where $p(x|s_i)$ is the posterior probability of x when observing the score s_i. Since $q(x)$ is Gaussian and the posterior has convenient Gaussian parts, manipulation of this yields an equation for μ_{new} and σ^2_{new} that can be solved using an iterative fixed-point approach:

Lemma 1. *Values for μ_{new} and σ^2_{new} minimizing KL$\left(q(x)||p(x|s_i)\right)$ satisfy*

$$\begin{aligned}\mu_{new} &= \sigma^2\left(s_i - e^{\kappa}\right) + \mu, \\ \sigma^2_{new} &= \frac{\sigma^2}{1 + \sigma^2 e^{\kappa}}, \end{aligned} \tag{15}$$

where

$$\kappa = \log\left(\frac{\mu + s_i\sigma^2 - 1 - \kappa + \sqrt{(\kappa - \mu - s_i\sigma^2 - 1)^2 + 2\sigma^2}}{2\sigma^2}\right). \tag{16}$$

Proof. The second term in (14) is evaluated using Bayes Theorem, $p(x|s_i) = p(s_i|x)p(x)/p(s_i)$. The term in $\log p(s_i)$ can be dropped because it is constant with respect to μ_{new} and σ^2_{new}. The term $E_{x\sim q(x)}[\log p(s_i|x)]$ is found by expanding the Poisson distribution and noting $E_{x\sim p(x)}[\exp(x)] = \exp(\mu + \sigma^2/2)$ (see the Supplemental material[3] for derivation). Thus it becomes

$$s_i\mu_{\text{new}} - \exp(\mu_{\text{new}} + \sigma^2_{\text{new}}/2) - \log(s_i!)\ . \tag{17}$$

The term $E_{x\sim q(x)}[\log p(x)]$ according to the derivation in the Supplemental material becomes

$$-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left(\sigma^2_{new} + \mu^2_{new} - 2\mu\mu_{new} + \mu^2\right)\ . \tag{18}$$

[3] Available at http://users.cecs.anu.edu.au/~sguo/sbsl_ecml2012_final_supple.pdf

Plugging (17) and (18) into (14) gives

$$\arg\min_{q(x)} \mathrm{KL}\left(q(x)||p(x|s_i)\right) \equiv \arg\min_{q(x)} -\log\sqrt{2\pi e\sigma^2_{new}} - \Bigg(\underbrace{s_i\mu_{\text{new}} - \exp(\mu_{\text{new}} + \sigma^2_{\text{new}}/2) - \log(s_i!)}_{E_{x\sim q(x)}(\log p(s_i|x))} \underbrace{-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left(\sigma^2_{new} + \mu^2_{new} - 2\mu\mu_{new} + \mu^2\right)}_{E_{x\sim q(x)}(\log p(x))} \Bigg).$$

To find the minimizer $q(x)$, we calculate the partial derivatives of $\mathrm{KL}\left(q(x)||p(x|s_i)\right)$ w.r.t. μ_{new} and σ_{new}, and set them to zero, leading to

$$\mu_{\text{new}} = \sigma^2\left(s_i - \exp\left(\mu_{\text{new}} + \frac{\sigma^2_{\text{new}}}{2}\right)\right) + \mu,$$

$$\sigma^2_{\text{new}} = \frac{\sigma^2}{1 + \sigma^2\exp(\mu_{\text{new}} + \frac{\sigma^2_{new}}{2})}\ .$$

Summing the first plus half the second of these equations, and defining $\kappa = \mu_{\text{new}} + \sigma^2_{\text{new}}/2$ yields the equation for κ of

$$\kappa = \mu + \sigma^2(s_i - \exp(\kappa)) + \frac{\sigma^2}{2(1 + \sigma^2\exp(\kappa))}, \tag{19}$$

and one gets (15) in terms of κ.

We convert (19) by solving for $\exp(\kappa)$ as it appears on the right-hand side. This yields a quadratic equation, and we take the positive solution since $\exp(\kappa)$ must be non-negative (see the Supplemental material). The result gives us (16).

We can use (16) as a fixed-point rewrite rule. For a given μ and σ^2 together with an initial value of κ, one iterates (16) until convergence. Empirically, this happens within 2-3 iterations. With convergence, we substitute the fixed-point solution into (15) to get the optimal mean and variance for $q(x)^*$.

Win Probability: Suppose we are given the offence and defence skills for team i and j, we can estimate the distributions over performance difference variables of x and y (c.f., Figure 2), and compute the Poisson parameters for s_i and s_j by using $\lambda_i = \exp(x)$ and $\lambda_j = \exp(y)$. To compute the winning probability of team i, i.e., $p(s_i > s_j)$, we first construct a new variable $s = s_i - s_j$, the difference variable between two Poisson distributions, which proves to be a Skellam distribution in [10]. Thus, we can compute the win probability of $P(s > 0)$ of team i, according to the probability mass function for the Skellam distribution

$$P(s = k; \lambda_i, \lambda_j) = e^{-(\lambda_i + \lambda_j)}\left(\frac{\lambda_i}{\lambda_j}\right)^{k/2} I_{|k|}\left(2\sqrt{\lambda_i\lambda_j}\right),$$

where $I_k(z)$ is the modified Bessel function of the first kind given in [1]. We approximated $P(s > 0, \lambda_i, \lambda_j)$ with $\sum_{k=1}^{n} P(s = k; \lambda_i, \lambda_j)$ using $n = 100$ since $P(s = k; \lambda_i, \lambda_j) \approx 0$ for all of our experiments when $k > 100$.

4.3 Inference in Gaussian-OD Model

Bayesian update: In the Gaussian-OD model (Figure 2(N)), all messages are Gaussian so one can compute the belief update in closed-form as follows

$$\pi_{o_i} = \frac{1}{\sigma_{o_i}^2} + \frac{1}{\beta_1^2+\beta_2^2+\gamma^2+\sigma_{d_j}^2}, \qquad \tau_{o_i} = \frac{\mu_{o_i}}{\sigma_{o_i}^2} + \frac{s_i+\mu_{d_j}}{\beta_1^2+\beta_2^2+\gamma^2+\sigma_{d_j}^2},$$
$$\pi_{d_j} = \frac{1}{\sigma_{d_j}^2} + \frac{1}{\beta_1^2+\beta_2^2+\gamma^2+\sigma_{o_i}^2}, \qquad \tau_{d_j} = \frac{\mu_{d_j}}{\sigma_{d_j}^2} + \frac{\mu_{o_i}-s_i}{\beta_1^2+\beta_2^2+\gamma^2+\sigma_{o_i}^2}, \tag{20}$$

where μ_{o_i} and σ_{o_i} are the mean and standard deviation of the prior offence skill distribution of team i, $\pi_{o_i}(\pi_{d_j}) = \frac{1}{\sigma^2_{post}}$ and $\tau_{o_i}(\tau_{d_j}) = \frac{\mu_{post}}{\sigma^2_{post}}$ are the precision and precision-adjusted mean for the posterior offence (defence) skill distribution of team i (j). Likewise, one can derive the update equations for team j's offence skill o_j and team i's defence skill d_i.

Win Probability: To compute the probability of team i winning vs team j, we first use message passing to estimate the normally distributed distributions for score variables s_i and s_j, and then compute the probability that $s_i - s_j > 0$, i.e., team i's score is larger than team j's. Given $s_i \sim \mathcal{N}(s_i; \mu_{si}, \sigma_{si}^2)$ and $s_j \sim \mathcal{N}(s_j; \mu_{sj}, \sigma_{sj}^2)$, we can compute the winning probability of team i by

$$p(s > 0) = 1 - \Phi\left(\frac{-(\mu_{si} - \mu_{sj})}{\sigma_{si}^2 + \sigma_{sj}^2}\right). \tag{21}$$

4.4 Inference in Gaussian-SD Model

Bayesian Update: In the Gaussian-SD model (Figure 3), all messages are Gaussian so we can again derive the update for the single team skills l_i and l_j in closed-form as follows:

$$\pi_{l_i} = \frac{1}{\sigma_{l_i}^2} + \frac{1}{\beta_1^2+\beta_2^2+\gamma^2+\sigma_{l_j}^2}, \qquad \tau_{l_i} = \frac{\mu_{l_i}}{\sigma_{l_i}^2} + \frac{(s_i-s_j)+\mu_{l_j}}{\beta_1^2+\beta_2^2+\gamma^2+\sigma_{l_j}^2}, \tag{22}$$
$$\pi_{l_j} = \frac{1}{\sigma_{l_j}^2} + \frac{1}{\beta_1^2+\beta_2^2+\gamma^2+\sigma_{l_i}^2}, \qquad \tau_{l_j} = \frac{\mu_{l_j}}{\sigma_{l_j}^2} + \frac{\mu_{l_i}-(s_i-s_j)}{\beta_1^2+\beta_2^2+\gamma^2+\sigma_{l_i}^2}, \tag{23}$$

where μ_{l_i} (μ_{l_j}) and σ_{l_i} (σ_{l_j}) are the mean and standard deviation of team i's (team j's) prior skill distribution, π_{l_i} (π_{l_j}) and τ_{l_i} (τ_{l_j}) are the precision and precision adjusted mean for team i's (team j's) posterior skill distribution.

Win Probability: To estimate the winning probability of team i for a match with team j, one can first use message passing to estimate the normally distributed score difference variable s, and then compute the winning probability of team i by

$$p(s > 0) = 1 - \Phi\left(\frac{l_i - l_j}{\sigma_i^2 + \sigma_j^2 + 2\beta^2}\right), \tag{24}$$

where l_i and σ_i are the mean and standard deviation for team i's skill level, and β the standard deviation of the performance variable.

5 Empirical Evaluation

5.1 Data Sets

Experimental evaluations are conducted on three data sets: Halo 2 XBox Live matches, Australian Football (Rugby) League (AFL)and UK Premier League (UK-PL)[4]. The Halo 2 data consists of a set of match outcomes comprising 6227 games for 1672 players. We note there are negative scores for this data, so we add the absolute value of the minimal score to all scores to use the data with all proposed models.

The training and testing settings are described as follows. For Halo 2 [5], the last 10% of matches are used for testing, and we use different proportions of the first 90% of data for training. There are 8 proportions used for training, ranging from 10% to 80% with an increment of 10%, and 90% is not used for training due to cross validation. To cross validate, we sample the data and run the learning 20 times at each proportion level to get standard error bars. Note that there are some players in the testing games who are not involved in any training data sets, particularly when small proportion of training data set is selected (e.g., the first 10 percent games); we remove these games in the testing set when reporting performances for all models.

For both UK-PL and AFL datasets, cross validation is performed by training and testing for each year separately (14 years for UK-PL, and 11 years for AFL). For these two datasets, we test the last 20% percent of matches in each year, with the training data increasing incrementally from 10% to 80% of the initial matches.

5.2 Evaluation Criteria

Information Gain. The first criterion we use to evaluate different approaches is *information gain*, which is proposed in the *Probabilistic Footy Tipping Competition*[6]: if a predictor assigns probability p to team i winning, then the score (in "bits") gained is $1 + \log_2(p)$ if team i wins, $1 + \log_2(1-p)$ if team i loses, $1 + (1/2)\log_2(p(1-p))$ if draw happens. In Section 4, we showed how to compute the win probability p for each model.

Win/No-Win Prediction Accuracy. While information gain provides a sense of how well the models fit the data, it is also interesting to see how accurate the models were at predicting match outcomes in terms of win/no-win (e.g., loss/draw). To compare classification performance of each model, we report the win/not winning prediction accuracy in terms of area under the curve (AUC) for the games with a win or loss outcome.

[4] http://www.football-data.co.uk/englandm.php

[5] Credit for the use of the Halo 2 Beta Data set is given to Microsoft Research Ltd. and Bungie.

[6] Refer to http://www.csse.monash.edu.au/~footy/

Score Prediction Error. We evaluate the score prediction accuracy for Poisson-OD and Gaussian-OD models for *each* team in terms of the mean absolute error (MAE). Note that we must omit the Gaussian-SD model since it can only predict score differences rather than scores.

5.3 Results

Experimental results are reported according to the parameter configurations shown in Table 1. All results are presented in Figure 4 and discussed below.

Table 1. Parameter settings. Priors on offence/defence skills: $\mathcal{N}(\mu_0, \sigma_0^2)$ with $\mu_0 = 25$ and $\sigma_0 = 25/3$. Performance variance: β, β_o, β_d.

Model	Parameter (ϵ, γ empirically estimated)
TrueSkill	$\beta = \sigma_0/2$, ϵ: draw probability
Poisson-OD	$\beta_o = \beta_d = \sigma_0/2$
Gaussian-OD	$\beta_o = \beta_d = \sigma_0/2$, γ: score variance
Gaussian-SD	$\beta = \sigma_0/2$, γ: score difference variance

Information Gain. For relatively small amounts of training data (10% – 30%), the Gaussian models (OD and SD) statistically significantly outperform TrueSkill and Poisson-OD in terms of win/loss probability accuracy. On all data sets except AFL, the Gaussian models perform comparably to TrueSkill for larger amounts of training data. Gaussian-OD statistically significantly outperforms Gaussian-SD for Halo 2, indicating that separate offence/defence modeling helps.

Win/No-Win Prediction Accuracy. In terms of win/no-win prediction accuracy, the Gaussian-OD model generally provides the best average AUC, followed by Gaussian-SD, then TrueSkill (with the exception of cases for Halo 2 with more than 40% training data where TrueSkill performs best), then Poisson-oD. Again, we see that the separate offence/defence skill modeling of Gaussian-OD gives it a performance edge over the combined skill model of Gaussian-SD.

Score Prediction Errors. As shown in the third column in Figure 4, Gaussian-OD predicts more accurate scores on the UK-PL and Halo datasets, while Poisson-OD is more accurate for the AFL dataset. This can be explained by a simple skill analysis — the learned skills on the UK-PL dataset tend to show a larger variance (for all models), whereas the learned skills on the AFL dataset show little variance (for all models except Gaussian-SD). Thus, the use of an exponentiated scoring rate in the Poisson-OD model would seem to amplify these small performance differences in learned AFL skills to make more accurate score predictions on AFL data. This amplification appears to hurt the Poisson-OD model on the lower-scoring UK-PL and Halo dataset (the mean score for the AFL data is 95.4 vs 42.7 and 1.3 respectively for the Halo 2 and UK-PL data).

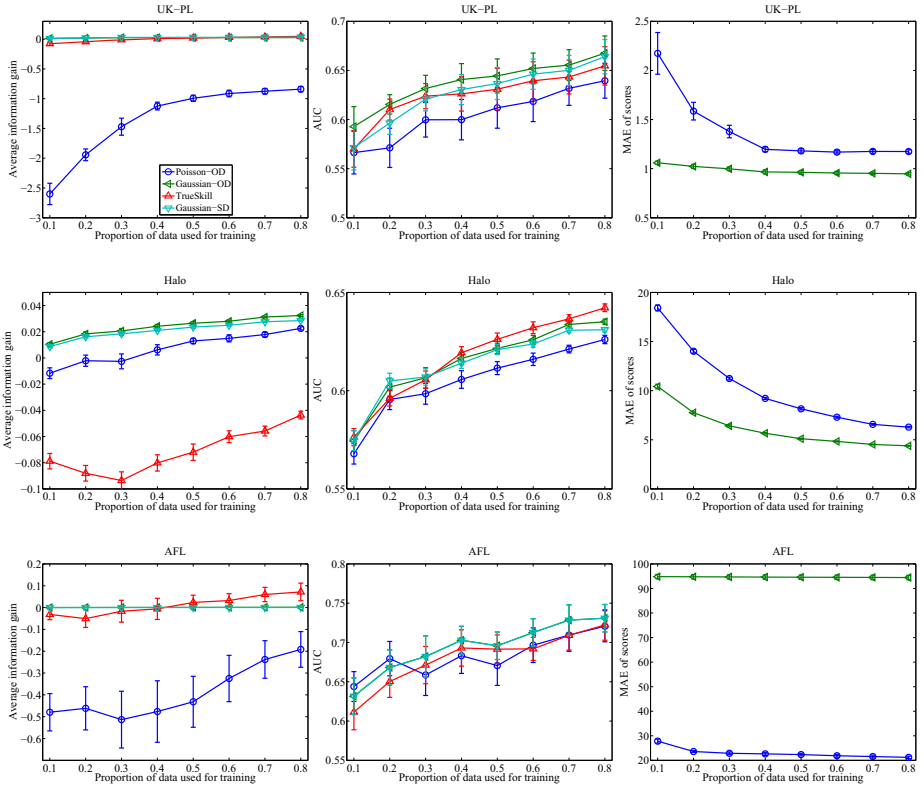


Fig. 4. Results on the UK-PL, Halo, and AFL datasets evaluated using information gain (left column), win/loss prediction accuracy in term of the area of the curve (AUC) (middle column), and score prediction error (right column). Error bars indicate 95% confidence intervals.

6 Related Work

Skill Rating dates at least as far back as the Elo system [4], the idea of which is to model the probability of the possible game outcome as a function of the two players' skill levels. Players' skill levels are updated after each game in a way such that the observed game outcome becomes more likely and the summation of players' ratings remains unchanged.

The Elo system cannot handle the case when three or more teams participate in one match, a disadvantage addressed by TrueSkill [5]. Further extensions of TrueSkill incorporate time-dependent skill modeling for historical data [3].

In [2], the authors model and learn the correlation between all players' skills when updating skill beliefs, and develop a method called "EP-Correlated", contrasted with the independent assumption on players' skills (EP-Independent). Empirically, EP-Correlated outperforms EP-Independent on professional tennis

match results; this suggests modeling correlations in extensions of the score-based learning presented here.

These skill learning methods all share a common feature that they are restricted to model WLD only and have to discard meaningful information carried with scores. While we proposed score-based extensions of TrueSkill in this work; it remains to incorporate other extensions motivated by this related work.

Score Modeling has been studied since the 1950s [15] [16] [11] [14] [13]; one of the most popular score models is the Poisson model, first presented in [15], and this work continues to the present [13]. Other commonly used score models are based on normal distributions [11]. However, it appears that most score-based models do not distinguish offence and defence skills of each team and the results here indicate that such separate offence/defence skill models can perform better than univariate models with limited data.

7 Conclusion

We proposed novel score-based, online Bayesian skill learning extensions of TrueSkill that modeled (1) player's offence and defence skills separately and (2) how these offence and defence skills interact to generate scores. Overall these new models — and Gaussian-OD (using a separate offence/defence skill model) in particular — show an often improved ability to model winning probability and win/loss prediction accuracy over TrueSkill, especially when the amount of training data is limited. This indicates that there is indeed useful information in score-based outcomes that is ignored by TrueSkill and that separate offence/defence skill modeling does help (c.f. the performance of Gaussian-OD vs. Gaussian-SD). Furthermore, these new models allow the prediction of scores (unlike TrueSkill), with the Poisson-OD model and its variational Bayesian update derived in Section 4.2 performing best on the high-scoring AFL data. Altogether, these results suggest the potential advantages of score-based Bayesian skill learning over state-of-the-art WLD-based skill learning approaches like TrueSkill.

Future research could combine the proposed models with related work that models home field advantage, time-dependent skills, multi-team games, and correlated skills to utilise score-based outcomes.

Acknowledgments. We thank Guillaume Bouchard and Onno Zoeter for interesting discussions, and we also thank the anonymous reviwers for their constructive comments, which help to improve the paper. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Abramowitz, M., Stegun, I.A.: Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables. Dover Publications, New York (1974)

2. Birlutiu, A., Heskes, T.: Expectation Propagation for Rating Players in Sports Competitions. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 374–381. Springer, Heidelberg (2007)
3. Dangauthier, P., Herbrich, R., Minka, T., Graepel, T.: Trueskill through time: Revisiting the history of chess. In: NIPS, pp. 337–344. MIT Press, Cambridge (2008)
4. Elo, A.E.: The rating of chess players: past and present. Arco Publishing, New York (1978)
5. Herbrich, R., Minka, T., Graepel, T.: Trueskill^{TM}: A Bayesian skill rating system. In: NIPS, pp. 569–576 (2006)
6. Karlis, D., Ntzoufras, I.: Bayesian modelling of football outcomes: using the skellam's distribution for the goal difference. IMA Journal of Management Mathematics 20(2), 133–145 (2009)
7. Kschischang, F.R., Frey, B.J., Loeliger, H.-A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47(2), 498–519 (2001)
8. Minka, T.: Expectation propagation for approximate bayesian inference. In: UAI, pp. 362–369. Morgan Kaufmann (2001)
9. Moroney, M.J.: Facts from figures, 3rd edn. Penguin Press Science (1956)
10. Skellam, J.G.: The frequency distribution of the difference between two Poisson variates belonging to different populations. Journal of the Royal Statistical Society: Series A 109(3), 296 (1946)
11. Glickman, M.E., Stern, H.S.: A state-space model for football league scores. Journal of the American Statistical Association 93(441), 25–35 (1998)
12. Beal, M.J., Ghahramani, Z.: The Variational Bayesian EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures. In: Proceedings of the Seventh Valencia International Meeting, pp. 453–464 (2002)
13. Karlis, D., Ntzoufras, I.: Bayesian modelling of football outcomes: using the Skellam's distribution for the goal difference. IMA Journal of Management Mathematics 20(2), 133–145 (2009)
14. Karlis, D., Ntzoufras, I.: Analysis of Sports Data by Using Bivariate Poisson Models. Journal of the Royal Statistical Society: Series D 52(3), 381–393 (2003)
15. Moroney, M.J.: Facts from figures, 3rd edn. Penguin Press Science (1956)
16. Dixon, M.J., Coles, S.G.: Modelling Association Football Scores and Inefficiencies in the Football Betting Market. Journal of the Royal Statistical Society: Series C 46(2), 265–280 (1997)

A Note on Extending Generalization Bounds for Binary Large-Margin Classifiers to Multiple Classes

Ürün Dogan[1], Tobias Glasmachers[2], and Christian Igel[3]

[1] Institut für Mathematik, Universität Potsdam, Germany
doganudb@math.uni-potsdam.de
[2] Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany
tobias.glasmachers@ini.ruhr-uni-bochum.de
[3] Department of Computer Science, University of Copenhagen, Denmark
igel@diku.dk

Abstract. A generic way to extend generalization bounds for binary large-margin classifiers to large-margin multi-category classifiers is presented. The simple proceeding leads to surprisingly tight bounds showing the same $\tilde{O}(d^2)$ scaling in the number d of classes as state-of-the-art results. The approach is exemplified by extending a textbook bound based on Rademacher complexity, which leads to a multi-class bound depending on the sum of the margin violations of the classifier.

1 Introduction

The generalization performance of binary (two-class) large-margin classifiers is well analysed (e.g., [1–8]). The theory of generalization bounds for multi-class support vector machines (multi-class SVMs) follows the route already paved by the analysis of the binary case, for instance, in the work of Guermeur [9, 10].

In this note, we link the analysis of binary and multi-class large margin classifiers explicitly. A straightforward technique to generalize bounds for binary learning machines to the multi-class case is presented, which is based on a simple union bound argument. The next section introduces the classification framework and extensions of large-margin separation to multiple classes. Section 3 proves our main result showing how to derive bounds for multi-category classification based on bounds for the binary case. In Section 4 we apply the proposed method to a textbook result based on Rademacher complexity. The newly derived bound is discussed with respect to different multi-class SVM formulations.

2 Large-Margin Multi-category Classification

We consider learning a hypothesis $h : X \to Y$ from training data

$$S = \big((x_1, y_1), \ldots, (x_\ell, y_\ell)\big) \in \big(X \times Y\big)^\ell \ ,$$

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 122–129, 2012.

where X and Y are the input and label space, respectively. We restrict our considerations to the standard case of a finite label space (however, there exist extensions of multi-class SVMs to infinite label spaces, e.g., [11]). We denote the cardinality $|Y|$ by $d \in \mathbb{N}$. Without loss of generality we assume $Y = \{1, \ldots, d\}$ in the sequel. We presume all data points (x_n, y_n) to be sampled i.i.d. from a fixed distribution P on $X \times Y$. Then the goal of learning is to map the training data to a hypothesis h with as low as possible risk (generalization error)

$$\mathcal{R}(h) = \int_{X \times Y} \mathbf{1}_{(h(x) \neq y)} \, \mathrm{d}P(x, y) \ . \tag{1}$$

Here, the 0-1 loss is encoded by the indicator function $\mathbf{1}_{(h(x)\neq y)}$ of the set $\{(x, y) \in X \times Y \,|\, h(x) \neq y\}$.

All machines considered in this study construct hypotheses of the form

$$x \mapsto \arg\max_{c \in Y} \left[\langle w_c, \phi(x) \rangle + b_c \right] \ , \tag{2}$$

where $\phi : X \to \mathcal{H}$ is a feature map into an inner product space $\mathcal{H}$, $w_1, \ldots, w_d \in \mathcal{H}$ are class-wise weight vectors, and $b_1, \ldots, b_d \in \mathbb{R}$ are class-wise bias/offset values. The most important case is that of a feature map defined by a positive definite kernel function $k : X \times X \to \mathbb{R}$ with the property $k(x, x') = \langle \phi(x), \phi(x') \rangle$. For instance, we can set $\phi(x) = k(x, \cdot)$, in which case $\mathcal{H}$ is the corresponding reproducing kernel Hilbert space [12]. We presume that the arg max operator in equation (2) returns a single class index (ties may, e.g., be broken at random). We define the vector-valued function $f : X \to \mathbb{R}^d$ by $f = (f_1, \ldots, f_d)$ with $f_c = \langle w_c, \phi(\cdot) \rangle + b_c$ for $c \in \{1, \ldots, d\}$. Then $h(x) = \arg\max_{c \in Y} f_c(x)$ and, to ease the notation, we define $\mathcal{R}(f)$ to be equal to the corresponding $\mathcal{R}(h)$.

For a binary classifier based on thresholding a real-valued function $f : X \to \mathbb{R}$ at zero we define the hinge loss $L^{\text{hinge}}(f(x), y) = \max\{0, 1 - y \cdot f(x)\}$, a convex surrogate for the 0-1 loss used in equation (1). The expression $y \cdot f(x)$ is the (functional) margin of the training pattern (x, y). The hinge loss measures the extent to which a pattern fails to meet a target margin of one. There are different ways to extend this loss to multiple classes. Large-margin classification based on the decision function (2) can be interpreted as highlighting differences between components $f_c(x)$. This is because the difference $f_c(x) - f_e(x)$ indicates whether class c is preferred over class e in decision making. Accordingly, two canonical extensions of the hinge loss to the multi-class case $f : X \to \mathbb{R}^d$ are the sum loss

$$L^{\text{sum}}(f(x), y) = \sum_{c \in Y \setminus \{y\}} \left[L^{\text{hinge}} \left(\frac{1}{2} \big(f_y(x) - f_c(x) \big), 1 \right) \right]$$

and the maximum loss

$$L^{\text{max}}(f(x), y) = \max_{c \in Y \setminus \{y\}} \left[L^{\text{hinge}} \left(\frac{1}{2} \big(f_y(x) - f_c(x) \big), 1 \right) \right] \ .$$

These losses are arranged so that in the binary case $d = 2$ they reduce to the hinge loss. We denote the corresponding risks by

$$\mathcal{R}^{\text{type}}(f) = \int_{X \times Y} L^{\text{type}}(f(x), y) \, \mathrm{d}P(x, y)$$

and the empirical risk for a sample $S = \big((x_1, y_1), \ldots, (x_\ell, y_\ell)\big)$ by

$$\mathcal{R}_S^{\text{type}}(f) = \frac{1}{\ell} \sum_{i=1}^{\ell} L^{\text{type}}(f(x_i), y_i) \ ,$$

where the superscript "type" is generic for "hinge", "sum", or "max".

3 Extending Bounds to Multi-category Classification

Our analysis relies on the basic insight that there are $d - 1$ distinct possible mistakes per example (x, y), namely preferring class $c \in Y \setminus \{y\}$ over the true class y. Each of these mistakes corresponds to one binary problem (having a decision function with weight vector $w_y - w_c$) indicating the specific mistake. One of these mistakes is sufficient for wrong classification, and no "binary" mistake at all implies correct classification. Then, a union bound over all mistakes gives the multi-class generalization result based on established bounds for binary classifiers. Assume that we have a bound of the following generic form:

Assumption 1. *With probability $1 - \delta$ over randomly drawn training sets S of size ℓ the risk $\mathcal{R}(f^{bin})$ of a binary classifier derived from a function $f^{bin} \in \mathbb{F}^{bin}$ is bounded by*

$$\mathcal{R}(f^{bin}) \leq B^{bin}\left(\ell, \mathcal{R}_S^{hinge}(f^{bin}), \mathcal{C}(\mathbb{F}^{bin}), \delta\right) \ ,$$

where $\mathbb{F}^{bin}$ is a space of functions $X \to \mathbb{R}$. The function $\mathcal{C}$ measures the complexity of the function class $\mathbb{F}^{bin}$ in a possibly data-dependent manner (i.e., it may implicitly depend on properties of the training data, typically in terms of the kernel Gram matrix).

Then we have:

Theorem 1. *Under Assumption 1, with probability $1 - \delta$ over randomly drawn training sets $S \in \big(X \times \{1, \ldots, d\}\big)^\ell$, the risk $\mathcal{R}(f)$ of a multi-class classifier derived from the function $f = (f_1, \ldots, f_d) : X \to \mathbb{R}^d$, $f \in \mathbb{F}$, using the decision rule (2) is bounded by*

$$\mathcal{R}(f) \leq \sum_{1 \leq c < e \leq d} \left(\frac{\ell^{(c,e)}}{\ell} + \frac{1}{\sqrt{\ell}} \sqrt{\frac{\log(d(d-1)) - \log \delta}{2}} \right) \cdot B^{bin}\left(\ell^{(c,e)}, \mathcal{R}_{S^{(c,e)}}^{hinge}\left(\frac{1}{2}(f_c - f_e)\right), \mathcal{C}(\mathbb{F}^{(c,e)}), \frac{\delta}{d(d-1)}\right) \ , \quad (3)$$

where $S^{(c,e)} = \left\{(x,y) \in S \,\middle|\, y \in \{c,e\}\right\}$ *is the training set restricted to examples of classes* c *and* e, $\ell^{(c,e)} = \left|S^{(c,e)}\right|$ *denotes its cardinality, and the pairwise binary function classes are defined as*

$$\mathbb{F}^{(c,e)} = \left\{ \frac{1}{2}(f_c - f_e) \,\middle|\, f = (f_1, \ldots, f_d) \in \mathbb{F} \right\} \quad .$$

Proof. Following the line of arguments above, the general case of d-category classification with $f = (f_1, \ldots, f_d) \in \mathbb{F}$ can be reduced to the binary case via the inequality

$$\begin{aligned} \mathcal{R}(f) &\leq \sum_{1 \leq c < e \leq d} [P(y=c) + P(y=e)] \cdot \mathcal{R}\left(\frac{1}{2}(f_c - f_e)\right) \\ &\leq \sum_{1 \leq c < e \leq d} \mathcal{R}\left(\frac{1}{2}(f_c - f_e)\right) \quad , \end{aligned}$$

where $\mathcal{R}\left(\frac{1}{2}(f_c - f_e)\right)$ refers to the risk of $\frac{1}{2}(f_c - f_e)$ solving the binary classification problem of separating class c from class e. Comparing the left to the right term of the above inequality for the risk gives the immediate result

$$\mathcal{R}(f) \leq \sum_{1 \leq c < e \leq d} B^{\text{bin}}\left(\ell^{(c,e)}, \mathcal{R}^{\text{hinge}}_{S^{(c,e)}}\left(\frac{1}{2}(f_c - f_e)\right), \mathcal{C}\left(\mathbb{F}^{(c,e)}\right), \frac{2\delta}{d(d-1)}\right) \quad ,$$

where we have split the probability δ over the samples into $d(d-1)/2$ equal chunks. This is conservative, since pairs of classes are highly dependent.

This bound can be refined by taking class-wise probabilities into account. By applying the Hoeffding bound we derive that $P(y=c) + P(y=e)$ is upper bounded by

$$\frac{\ell^{(c,e)}}{\ell} + \frac{1}{\sqrt{\ell}} \sqrt{\frac{\log(d(d-1)) - \log\delta}{2}}$$

with a probability of $1 - \delta'/2$ with $\delta' = 2\delta/(d(d-1))$. That is, this bound holds simultaneously for all $d(d-1)/2$ pairs of classes with a probability of $1 - \delta/2$. □

The pairwise complexity terms $\mathcal{C}(\mathbb{F}^{(c,e)})$ can be replaced with the complexity measure

$$\mathcal{C}(\mathbb{F}) = \max_{1 \leq c < e \leq d} \mathcal{C}\left(\mathbb{F}^{(c,e)}\right)$$

for $\mathbb{R}^d$-valued functions. Depending on the structure of the underlying binary bound B^{bin} the sum over all pairs of classes can be further collapsed into a factor of $d(d-1)/2$, for instance by taking the maximum over the summands.

4 Example: A Bound Based on Rademacher Complexity

Theorem 1 can be used to obtain a variety of generalization bounds when combined with the wealth of results for binary machines that can be brought in a form of Assumption 1. This section will consider an textbook generalization bound derived for binary SVMs and measuring function class flexibility by Rademacher complexity. The result will then be discussed w.r.t. two multi-class extensions of SVMs. Such a comparison can be performed with different goals in mind. On the one hand, a unifying analysis covering many different multi-class SVM types is desirable. On the other hand, one would like to see differences in the performance guarantees for different machines that may indicate superiority of one machine over another. We attempt to highlight such differences. This is in contrast to other studies such as the influential work in [9], where the goal was unification and, therefore, to make differences between machines invisible.

4.1 Extending a Binary Bound Based on Rademacher Complexity

We begin by stating the result for binary machines, in which the Rademacher complexity of real-valued functions is bounded based on the kernel Gram matrix K of the data:

Theorem 2. *Fix $\rho > 0$ and let $\mathbb{F}^{bin}$ be the class of functions in a kernel-defined feature space with norm at most $1/\rho$. Let S be a training set of size ℓ, and fix $\delta \in (0,1)$. Then with probability of at least $1-\delta$ over samples of size ℓ we have for $f^{bin} \in \mathbb{F}^{bin}$*

$$\mathcal{R}(f^{bin}) \leq B^{bin} = \mathcal{R}_S^{hinge}(f^{bin}) + \frac{4}{\ell\rho}\sqrt{\operatorname{tr}(K)} + 3\sqrt{\frac{\log(2/\delta)}{2\ell}} \ ,$$

where K is the kernel Gram matrix of the training set.

This result can be derived following the proof of Theorem 4.17 by [3]. Application of inequality (3) yields the following generalization bound:

Corollary 1. *Let $S \in \big(X \times \{1,\dots,d\}\big)^\ell$ be a training set. Fix $\rho > 0$, and let $\mathbb{F}_\rho$ be the class of $\mathbb{R}^d$-valued functions in a kernel-defined feature space with semi-norm at most $1/\rho$ w.r.t. the semi-norm $\|f\| = \max\left\{\frac{1}{2}\|f_c - f_e\| \,\middle|\, 1 \leq c < e \leq d\right\}$. With probability $1-\delta$ over randomly drawn training sets $S \in \big(X \times \{1,\dots,d\}\big)^\ell$, the risk $\mathcal{R}(f)$ of a multi-class classifier using the decision rule (2) is bounded by*

$$\mathcal{R}(f) \leq \sum_{1 \leq c < e \leq d} \left(\frac{\ell^{(c,e)}}{\ell} + \frac{1}{\sqrt{\ell}} \sqrt{\frac{\log(d(d-1)) - \log\delta}{2}} \right) \cdot \left[\mathcal{R}^{hinge}_{S^{(c,e)}} \left(\frac{1}{2}(f_c - f_e) \right) + \frac{4}{\ell^{(c,e)}\rho} \sqrt{\operatorname{tr}(K^{(c,e)})} + 3\sqrt{\frac{\log(2d(d-1)/\delta)}{2\ell^{(c,e)}}} \right] \ ,$$

where $K^{(c,e)}$ denotes the $\ell^{(c,e)} \times \ell^{(c,e)}$ kernel matrix restricted to examples of classes c and e.

With $\mathrm{tr}(K^{(c,e)}) \leq \mathrm{tr}(K)$ this bound reads in (not fully simplified) $\tilde{O}$-notation

$$\mathcal{R}(f) \in \tilde{O}\left(\frac{d(d-1)}{2}\left(\frac{4}{\rho \cdot \ell} \cdot \sqrt{\mathrm{tr}(K)} + \mathcal{R}_S^{\mathrm{sum}}(f)\right)\right) \quad , \tag{4}$$

with the same separation of complexity and empirical risk terms as in the binary bound.[1]

4.2 Sum vs. Maximum of Margin Violations

There is no canonical extension of the binary SVM [13, 14] to multiple classes. Several slightly different formulations have been proposed, most of which reduce to the standard binary SVM if $d = 2$.

The all-in-one methods proposed independently Weston and Watkins [15], Vapnik ([1], Section 10.10), and by Bredensteiner and Bennett [16] turned out to be equivalent, up to rescaling of the decision functions and the regularization parameter $C > 0$. The method is defined by the optimization problem

$$\min \quad \frac{1}{2}\sum_{c=1}^{d} \langle w_c, w_c \rangle + C \cdot \mathcal{R}_S^{\mathrm{sum}}(f) \quad .$$

An alternative multi-class SVM was proposed by Crammer and Singer [17]. It also takes all class relations into account simultaneously and solves a single optimization problem, however, penalizing the maximal margin violation instead of the sum:

$$\min \quad \frac{1}{2}\sum_{c=1}^{d} \langle w_c, w_c \rangle + C \cdot \mathcal{R}_S^{\max}(f)$$

Are there theoretical arguments to prefer one formulation over the other? In [18], the *empirical* risk of multi-class SVMs is upper bounded in terms of the empirical maximum risk $\mathcal{R}^{\max}$. This is an almost trivial result, because the hinge loss (and therefore the maximum loss) is, per construction, an upper bound on the 0-1-loss. Based on this bound it has been argued that the SVM proposed by Crammer and Singer has advantages compared to the formulation by Weston and Watkins because it leads to lower values in the bounds.

We do not find this argument convincing. The empirical error is only a weak predictor of the generalization error, and measuring these errors with different loss functions is a meaningless comparison. The question which hypothesis has lower 0-1-risk cannot be decided on this basis, but only by comparing generalization bounds.

When looking at the bound newly derived above we find that it depends on the sum-loss term $\mathcal{R}_S^{\mathrm{sum}}(f)$. Thus, one may argue that it is a *natural* strategy to minimize this quantity directly instead of the max-loss.

[1] The $\tilde{O}$ (soft O) notation ignores logarithmic factors, not only constant factors. That is, $f(\ell) \in \tilde{O}(g(\ell))$ iff $f(\ell) \in O(g(\ell)\log^{\kappa} g(\ell))$ for some κ.

In general, comparing different machines by means of generalization bounds can be misleading for a number of reasons. The most important is that we are only dealing with upper bounds on the performance, and a *better performance guarantee* does give a *guarantee for better performance*.

5 Discussion

The proposed way to extend generalization bounds for binary large-margin classifiers to large-margin multi-category classifiers is very simple, compared to taking all pairwise interactions between classes into account at once, and it has a number of advantageous properties. It is versatile and generic in the sense that it is applicable to basically every binary margin-based bound. Compared to the underlying bounds we pay the price of considering the worst case over $d(d-1)/2$ pairs of classes. However, also the state-of-the-art results obtained in [9] exhibit the same $\tilde{O}(d^2)$ scaling in the number of classes. In any case this term does not affect the asymptotic tightness of the bounds w.r.t. the number of samples. The same argument, put the other way round, implies that the asymptotic tightness of a bound for binary classification carries over one-to-one to the multi-class case. This implies that binary and multi-class learning have the same sample complexity.

It is straightforward to extend our result to loss functions based on general confusion matrices. Future work may include applying the proposed procedure to more sophisticated bounds for binary classifiers.

Acknowledgements. Christian Igel gratefully acknowledges support from the European Commission through project AKMI (PCIG10-GA-2011-303655).

References

1. Vapnik, V.: Statistical Learning Theory. John Wiley and Sons (1998)
2. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2002)
3. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
4. Boucheron, S., Bousquet, O., Lugosi, G.: Theory of classification: A survey of some recent advances. ESAIM: Probability and Statistics 9, 323–375 (2005)
5. Bartlett, P.L., Jordan, M.I., McAuliffe, J.D.: Convexity, classification, and risk bounds. Journal of the American Statistical Association 101, 138–156 (2006)
6. Wu, Q., Ying, Y., Zhou, D.X.: Multi-kernel regularized classifiers. Journal of Complexity 23, 108–134 (2007)
7. Steinwart, I., Scovel, C.: Fast rates for support vector machines using Gaussian kernels. The Annals of Statistics 35, 575–607 (2007)
8. Steinwart, I.: Oracle inequalities for svms that are based on random entropy numbers. Journal of Complexity 25, 437–454 (2009)
9. Guermeur, Y.: VC theory for large margin multi-category classifiers. Journal of Machine Learning Research 8, 2551–2594 (2007)

10. Guermeur, Y.: Sample complexity of classifiers taking values in $\mathbb{R}^Q$, Application to multi-class SVMs. Communications in Statistics: Theory and Methods 39, 543–557 (2010)
11. Bordes, A., Usunier, N., Bottou, L.: Sequence Labelling SVMs Trained in One Pass. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 146–161. Springer, Heidelberg (2008)
12. Aronszajn, N.: Theory of reproducing kernels. Transactions of the American Mathematical Society 68, 337–404 (1950)
13. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT 1992), pp. 144–152. ACM (1992)
14. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20, 273–297 (1995)
15. Weston, J., Watkins, C.: Support vector machines for multi-class pattern recognition. In: Verleysen, M. (ed.) Proceedings of the Seventh European Symposium On Artificial Neural Networks (ESANN), pp. 219–224. d-side Publications, Belgium (1999)
16. Bredensteiner, E.J., Bennett, K.P.: Multicategory classification by support vector machines. Computational Optimization and Applications 12, 53–79 (1999)
17. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research 2, 265–292 (2002)
18. 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)

Extension of the Rocchio Classification Method to Multi-modal Categorization of Documents in Social Media

Amin Mantrach and Jean-Michel Renders

Yahoo! Research Barcelona*, Xerox Research Centre Europe
amantrac@yahoo-inc.com, jean-michel.renders@xrce.xerox.com

Abstract. Most of the approaches in multi-view categorization use early fusion, late fusion or co-training strategies. We propose here a novel classification method that is able to efficiently capture the interactions across the different modes. This method is a multi-modal extension of the Rocchio classification algorithm – very popular in the Information Retrieval community. The extension consists of simultaneously maintaining different "centroid" representations for each class, in particular "cross-media" centroids that correspond to pairs of modes. To classify new data points, different scores are derived from similarity measures between the new data point and these different centroids; a global classification score is finally obtained by suitably aggregating the individual scores. This method outperforms the multi-view logistic regression approach (using either the early fusion or the late fusion strategies) on a social media corpus - namely the ENRON email collection - on two very different categorization tasks (folder classification and recipient prediction).

1 Introduction

Multi-modal (or multi-view[1]) learning has been intensively studied since a long period. It relates to the problem of learning from multiple set of features. As suggested by [1], the multi-modal learning takes its justification from the fact that a high consensus of two independent hypotheses results in a low generalization error. These last years, several new methods have been proposed for the semi-supervised and the unsupervised settings (for semi-supervised multi-view learning, see the survey in [2]). However, few has been done in the fully supervised setting.

Actually, as pointed out by [3], in a fully supervised setting, multi-modal learning usually performs worse than learning on the union of all modes. In this setting, the standard approaches are the *early fusion* (EF) method and the *late fusion* (LF) method. The former consists of directly learning from a common

* This work has been done when the author was with Xerox Research Centre Europe.
[1] The multi-view learning has different designations in the literature depending on the communities and the time period. In this paper, "views" and "modes" are considered as synonyms referring to the same concept.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 130–142, 2012.

global view which is made from the union of all mono-views while the latter proposes to combine the decisions of different single-view based classifiers. On the one hand, the principle of the *EF* strategy is to directly learn the combination of all the features that minimizes a loss function. The drawback of this approach is to combine artificially different data sources (with possibly different semantics), hence increasing the dimension of the feature space, which could result in a higher variance of the generalization error. On the other hand, the goal of *LF* strategy is to obtain a consensus among independent specialized classifiers leading to a lower generalization error. The drawback of this approach lies in its inability to detect interactions; in other words, it miss the opportunity to capture correlations between different views which may also lead to a lower generalization error and a better understanding of the underlying data.

In this work, we propose a novel multi-modal (*MM*) framework that tries to exploit the best of the two strategies, while avoiding their drawbacks. It offers the *EF*'s advantage of capturing interactions across the different modes while, in the meantime, owning the *LF*'s advantage of learning a weighted combination of the input scores in order to, first, detect a high consensus among the hypotheses and, second, have a better understanding of interactions between views. To achieve this goal, we propose to extend the standard "Rocchio" classification algorithm (see [4]) to the multi-modal case by computing "cross-modal" scores that measure the interaction between pairs of modes. In a nutshell, the "Rocchio" classification algorithm builds prototypes (centroids) for each class and classifies a new instance by linearly combining the similarity of this instance with the class prototypes. In the same vein, we propose in this work to extend the notion of (mono-modal) class prototype by defining for each class: (1) one (mono-modal) centroid per mode (which corresponds to the prototypes defined in the standard "Rocchio" classification algorithm) and (2) two "cross-modal" centroids per pair of modes. While simple centroids aim at focusing on the mono-modal aspects of the data, "cross-modal" centroids bias one mode by using the other one and hence take into account the multi-modal aspects of the data. When classifying a new input, it is compared with these different centroids (mono- and cross-modal) using a similarity function (for instance, the cosine or any similarity measure of information retrieval). These similarity scores form then the inputs of a linearly weighted *LF* process.

This novel *MM* framework is benchmarked on a social media corpus – namely the ENRON corporate email data set – on two different tasks: a foldering task and a recipient proposal task. For the foldering task, we consider the seven mailboxes selected by [5]. These mailboxes have been intensively benchmarked due to their public availability. Our framework is compared to *EF* and *LF* which constitute the state-of-the art for these tasks on this data set (see for instance the recent work of [6]). For the recipient proposal task, we evaluate the proposed framework on three mailboxes among the ones having the largest amount of messages.

We show that, thanks to the introduced cross-modal scores, our *MM* framework outperforms the state-of-the-art on this public data set for both tasks.

To summarize, the main contributions of this paper are the following:

- It introduces a new competitive framework which can be used for classification in case of multi-modal inputs.
- It proposes to compute cross-modal centroids which reflect the multi-modal aspects of the data.
- It shows on various mailboxes of the ENRON data set that the proposed method outperforms established methods, i.e. the *EF* and the *LF*, on foldering and recipient proposal tasks.

2 Problem Statement

More concretely, the multi-modal classification problem may be formalized as follows: We have at our disposal a collection of N_d data instances represented by M different data matrices $\mathbf{X}^{(m)}$ of size $N_d \times N_f^{(m)}$, where $N_f^{(m)}$ is the number of features associated with the mode $m \in \mathcal{M}$ (the set of modes $\{1, ..., M\}$). The vector $\mathbf{x}_{d\bullet}^{(m)}$ denotes the row d of $\mathbf{X}^{(m)}$ and $x_{i,j}^{(m)}$ denotes the entry (i, j) of the same matrix. $\mathbf{X}$ is the matrix obtained by concatenating the different $\mathbf{X}^{(m)}$ matrices in an ascending mode order, i.e. $\mathbf{X} = [\mathbf{X}^{(1)}\mathbf{X}^{(2)}...\mathbf{X}^{(|M|)}]$. The vector $\mathbf{x}_{d\bullet}^{(\bullet)}$ denotes the row d of this matrix.

For learning, we have a set of labeled multi-modal data instances $\mathcal{L} = \{\mathbf{x}_{d\bullet}^{(\bullet)}, y_d\}, d \in [1, N_d], y_d \in \mathcal{C}, \mathbf{x}_{d\bullet}^{(\bullet)} \in \mathcal{X}$. $\mathcal{C}$ is the set of the different labels: $\{c_1, ..., c_{|\mathcal{C}|}\}$ and $\mathcal{X}$ the set of multi-modal data instances.

We want to design a model $M_c \in \mathcal{M}$ (the set of multi-modal models) for each class c $\in \mathcal{C}$ and a multi-modal classification function $F : \mathcal{X} \times \mathcal{M} \rightarrow \mathbb{R}$. The predicted class of an unlabeled multi-modal point $\mathbf{x}_{u\bullet}^{(\bullet)}$ will be $\widehat{y} = \text{argmax}_c F(\mathbf{x}_{u\bullet}^{(\bullet)}, M_c)$.

One way to solve the problem consists of learning the classification models M_c's directly in the feature space which consists of the union of all modes. In this case, we are using an early fusion (*EF*) approach. However, as suggested by [7], in order to achieve good performance, each mode should be normalized separately before combination. This is mostly to make features comparable with respect to the range of values across the different modes and features.

Another standard approach consists of learning separately the different classification models M_c^m's for each mode m. After learning, the decision function consists of a linear combination of the different scores obtained on each mode m.

$$\widehat{y} = \text{argmax}_c \sum_m \alpha_c^m F(\mathbf{x}_{u\bullet}^{(m)}, M_c^m) \tag{1}$$

The weights α_c^m attributed to the different modes for each class are generally tuned on a independent validation set. This method is called late fusion (*LF*) as opposed to early fusion. Note that most of the proposed and widely used learning to rank systems are based on the linear combination of features [8, 9].

The advantage of the early fusion lies in its ability to capture relations between features across different modes. Furthermore, it avoids the supplementary

task of tuning hyper-parameters needed in the case of a late fusion approach. However, the early fusion suffers from combining artificially into one vector space multiple sources having different semantics. Furthermore, the level of sparsity may change across modes. Indeed, for instance, image features are dense, while bag-of-words document vectors are sparse and social-media participant vectors are binary sparse vectors. Another drawback of the early fusion method is to increase drastically the dimension of the feature space which results in increasing the variance of the generalization error. Hence, the late fusion strategy which consists of combining different expert models results generally in a lower variance. Its drawback is its inability to capture any possible combination of features across different modalities that may result in a better prediction.

In the remainder, we introduce a new framework which takes a decision based on a linear combination of multiple experts and therefore is LF-based. However, while the standard LF does not take into account the interactions across the multiple modes, this framework proposes to exploit the multi-modal aspects of the data. The proposed classification procedure, inspired by the trans-media relevance feedback in information retrieval [10], consists of the following steps: (1) off-line modeling of each class with a representative centroid per mode m obtained after aggregating all the mode m portion of the data instances, (2) off-line modeling of each class with two representative centroids per pair of modes (i.e. $(m)[m']$ and $(m')[m]$) obtained after aggregating the mode m portion of the nearest neighbors computed through the mode m' portion of the data instances, (3) defining similarity scores between unlabeled data points and the different centroids, (4) late (linear) fusion of the obtained scores in order to get a global membership score for each class and (5) applying a simple argmax function on global scores computed for each class in order to predict the class $\widehat{y}$.

3 Multi-modal Classes Modeling

Mono-modal Modeling. The off-line training phase consists of learning one model M_c for each class c. We propose to build a multi-modal centroid vector for each class c which summarizes the subset $\mathcal{L}_c$ of all the labeled data instances $\mathcal{L}$ which have class c as label. In other words, $\mathcal{L}_c = \{\mathbf{x}_{d\bullet}^{(\bullet)}, c\}$, is the set of all data instances that belongs to class c. Each centroid is made by aggregating all data instances d of $\mathcal{L}_c$.

$$C^{(m)}(c) = \bigoplus_{d \in \mathcal{L}_c} \frac{\mathbf{x}_{d\bullet}^{(m)}}{w_d} \tag{2}$$

$\bigoplus$ is an aggregation operator which may be an average, a max, a min. w_d is a weighting factor which counts the number of classes the document d belongs to. Concretely, the intuition is that in a multi-label setting, as for instance in the recipient proposal task (see Section 5.2), a multi-labeled document is less representative of the classes it belongs to than a mono-labeled document considered as more specific to the class.

Cross-modal Modeling. So far, the centroids are mono-modal and can only capture the modality in which they are defined. Therefore, following our main goal to capture interactions across modes, we define "cross-centroids". A cross-centroid in mode m is made by aggregating the m-mode portion of different nearest neighbors observations $\mathbf{x}_{d'\bullet}^{(m)}$. The nearest-neighbors are chosen in $\mathcal{L}$ according to a specific mode m' resulting in the cross-centroid $C^{(m)[m']}(c)$. Notice that, m and m' may be equal re-enforcing then the mono-modality aspect of the centroid. More formally, a cross-centroid is computed as follows:

$$C^{(m)[m']}(c) = \bigoplus_{d' \in \mathrm{NN}(\mathbf{x}_{d\bullet}^{(m')}), d \in \mathcal{L}_c} \mathbf{x}_{d'\bullet}^{(m)} w(\mathbf{x}_{d\bullet}^{(m')}, \mathbf{x}_{d'\bullet}^{(m')}) \tag{3}$$

where $NN(x^{(m')})$ is the "nearest neighbors" function returning the set of nearest neighbors of x in the mode m' and using any traditional similarity measure of information retrieval, the function $w(\mathbf{x}_{d\bullet}^{(m')}, \mathbf{x}_{d'\bullet}^{(m')})$ gives a weight proportional to the (mono-modal) similarity between both elements.

4 Multi-modal Late Fusion

To assess the affinity of an unlabeled observation to a specific class c, we compute its global similarity with the different centroids (i.e. mono modal and cross-modal) representing each class. The multi-modal unlabeled input, $\mathbf{x}_{u\bullet}^{(\bullet)}$, is compared mode by mode with the different centroids. Then, a global membership score is deduced from a linear combination of each mode contribution. The global score is thus computed as follows:

$$\begin{aligned} RSV(c, \mathbf{x}_{u\bullet}^{(\bullet)}) = & \sum_m \alpha_{c,m} \, \mathrm{sim}(C^{(m)}, \mathbf{x}_{u\bullet}^{(m)}) \\ & + \sum_m \sum_{m'} \beta_{c,m,m'} \, \mathrm{sim}(C^{(m)[m']}, \mathbf{x}_{u\bullet}^{(m)}) \end{aligned} \tag{4}$$

where $\alpha_{c,m}$ and $\beta_{c,m,m'}$ are positive weights summing up to 1. The "sim" function may be any traditional similarity measure used in information retrieval. The first part of Equation 4 denotes the simple sum of the similarities of each mode. This part does not cover any interaction across the modes. The last part of the equation aims at capturing the interactions across the different modes by computing the similarity with cross-modal centroids. In the remainder, we will show empirically that computing these cross-modal similarities lead to better performance for multi-modal categorization tasks. As with late fusion, the hyper-parameters can be learned in order to maximize a specific utility function (by cross-validation), for example: precision@1 or NDCG@10.

5 Experiments and Discussions

5.1 The ENRON Data Set

The introduced multi-modal (*MM*) framework is validated on the ENRON dataset. This dataset consists of a set of vectors and matrices that represent the whole ENRON corpus [see, e.g., 11], after linguistic preprocessing and metadata extraction. The linguistic preprocessing consists of removing some particular artefacts of the collection (for instance some recurrent footers, that have nothing to do with the original collection but indicate how the data were extracted), removing headers for emails (From/To/... fields), removing numerals and strings formed with non-alphanumeric characters, lowercasing all characters, removing stopwords as well as words occurring only once in the collection. There are two types of documents: documents are either (parent) emails or attachments (an attachment could be a spreadsheet, a power-point presentation, ...; the conversion to standard plain text is already given by the data provider). The ENRON collection contains 685,592 documents (455,449 are parent emails, 230,143 are attachments) extracted from 151 different mailboxes. We decided to process the attachments simply by merging their textual content with the content of the parent email, so that we have to deal only with parent emails. For parent emails, we have not only the content information, but also metadata. The metadata consist of:

- the custodian (i.e. the person who owns the mailbox from which this email is extracted);
- the location (i.e. the folder decomposition of each mailbox owner);
- the date or timestamp;
- the Subject field (preprocessed in the same way as standard content text);
- the From field;
- the To field ;
- the CC field.

After preprocessing, we summarize the data by two views for each data point: the textual view and the social view. The textual view consists of a bag-of-word representation of the aggregation of the "Body" vector, the "Attachment" vector and "Subject" vector. The social view is a vector in the bag-of-participants space which is the aggregation of the "From" vector, the "To" vector and the "Cc" vector.

For the foldering task, among the 151 available mailboxes, we consider 7 mailboxes having a sufficient amount of folders selected in [5]. Furthermore, we removed folders which are not specific to the considered mailbox but which have been automatically generated by an email client software. For instance, the folders "All documents", "Calendar", "Sent mail", "Deleted Items", "Inbox", "Sent Items", "Unread Mail", "Contacts" and "Drafts" have been discarded from the evaluation. Statistics on the seven selected folders are reported in Table 1. For the recipient proposal task, we report results for three mailboxes among the five having the largest amount of emails in the collection, namely: Vince J. Kaminski, Jeff Dasovich and Tana Jone's mailboxes.

Table 1. Folders distribution among of the preprocessed mailboxes used for the foldering task

Mailbox	#Folders	Total #em	Min #em.	Max #em.	Aver # em/Folder
Beck	60	258	1	27	4.30
Farmer	27	1689	1	528	62.56
Kaminski	32	842	1	120	26.31
Kitchen	51	4162	1	784	81.61
Lokay	14	1837	1	915	131.21
Sanders	20	411	2	181	20.55
Williams	21	2043	1	1076	97.29

5.2 Benchmark Protocol

The proposed model is tested on two different mail management tasks: (1) email foldering whose goal is to retrieve the correct folder in which an email should go using all its information (i.e. textual content and social metadata), (2) recipient proposal whose goal is to automatically propose recipient candidates for a new written message based only on the textual part of the email. It is important to note that the temporal aspect plays here a big role. Indeed, it would be generally easier to predict the recipient of new emails based on recent posts than on old ones. This comes from the fact that generally emails are part of a global discussion thread. In case of foldering, users may create folders, delete or move emails after an amount of time, for instance to the "Trash" folder to free memory on the server. We could introduce the temporal aspect in our model when building centroids by weighting the contribution of each message using the timestamp meta data. However, in order to compare more fairly with state-of-the-art *LR* and *ER* fusion methods we decide to not include the temporal information directly into the model. Instead, we decide to make training-test splits which reflect the sequential aspect of the data such that the compared methods are tested on more recent emails than those used during the training phase. Hence, we assume that the emails of the collection are sorted by increasing order of their timestamp.

For the foldering task, we consider training sets composed by 50% of the mailbox. After learning, the goal is to predict the folder of the next 10% emails in the temporal sequence. For example, when considering a training set made from the first 50% of the collection (i.e. in terms of timestamp), the test set then consists of the emails in the interval 50%-60%. By time-shifting the training and the test sets by 10%, we may consider training on the emails going from 10% to 60% and test on the next 10%, and so on. So that, finally, we define 5 possible training and test sets. The different methods require some hyper-parameters to be set. For this purpose, during each training phase, hyper-parameters are tuned internally by dividing the 50% training set into 40% internal-training and 10% internal-test. For the recipient proposal task, we consider training sets and test sets made each by 10% of the mailbox. By time-shifting by 10% the training-test sets we define 9 different splits. The first two splits are used as validation set for

Table 2. Table presenting the results for a foldering task averaged on 5 chronological ordered training-test pairs for the Beck's, Farmer's, Kaminski's and Kitchen's mailboxes. The state-of-the art methods late fusion (*LF*) and early fusion (*EF*) are compared with the proposed multi-modal framework. The reported performance measure for this mono-label classification task are the Recall@1,@5, @10, the NDCG@10 and the NDCG. The hyper-parameters of each algorithm has been tuned on an independent validation set.

Alg.	R@1	R@3	R@5	R@10	NDCG@10	NDCG
Beck's mailbox						
MM	**0.55** +/- 0.15	**0.68** +/- 0.12	**0.71** +/- 0.08	**0.77** +/- 0.11	**0.66** +/- 0.11	**0.68** +/- 0.10
LF	0.42 +/- 0.1	0.57 +/- 0.06	0.62 +/- 0.06	0.70 +/- 0.04	0.56 +/- 0.06	0.60 +/- 0.06
EF	0.46 +/- 0.14	0.63 +/- 0.15	0.65 +/- 0.11	0.69 +/- 0.09	0.59 +/- 0.12	0.63 +/- 0.10
Farmer's mailbox						
MM	0.68 +/- 0.11	**0.82** +/- 0.09	**0.87** +/- 0.08	**0.89** +/- 0.07	**0.79** +/- 0.08	**0.81** +/- 0.08
LF	0.65 +/- 0.14	0.79 +/- 0.13	0.81 +/- 0.12	0.86+/- 0.09	0.76 +/- 0.12	0.78 +/- 0.11
EF	0.68 +/- 0.13	0.77 +/- 0.12	0.81 +/- 0.11	0/85 +/- 0.08	0.76 +/- 0.11	0.78 +/- 0.10
Kaminski's mailbox						
MM	**0.54** +/- 0.20	**0.68** +/- 0.21	**0.73** +/- 0.16	**0.82** +/- 0.18	**0.67** +/- 0.19	**0.70** +/-0.18
LF	0.44 +/- 0.19	0.61 +/- 0.25	0.67 +/- 0.22	0.77 +/- 0.17	0/60 +/- 0.19	0.63 +/- 0.19
EF	0.51 +/- 0.22	0.61 +/- 0.21	0.67 +/- 0.23	0.78 +/- 0.21	0.63 +/- 0.21	0.66 +/- 0.20
Kitchen's mailbox						
MM	0.42 +/- 0.13	0.65 +/- 0.07	0.77 +/- 0.07	0.87 +/- 0.08	0.63 +/- 0.09	0.65 +/- 0.08
LF	0.41 +/- 0.12	0.65 +/- 0.16	0.75 +/- 0.13	0.81 +/- 0.14	0.61 +/- 0.13	0.64 +/- 0.12
EF	0.44 +/- 0.12	0.64 +/- 0.12	0.74 +/- 0.10	0.83 +/- 0.10	0.63 +/- 0.11	0.66 +/- 0/09

tuning the hyper-parameters of the different methods. The remaining splits are used for assessing the performance of the different algorithms.

5.3 Classification Models and Tuning

The benchmarked classification models are (1) the *MM* model proposed in this paper, the (2) *EF* model and the (3) *LF* model. For the *EF* and *LF* fusion models we use a one-vs-rest logistic regression classifier with a l2-norm regularization. This provides, after normalization, the posterior probability of each class given a test data point. As previously said, the regularization parameter is internally tuned during training for the foldering task, or tuned on an independent validation set for the recipient proposal task. For the *LF* strategy, the best convex combination (i.e. achieving the best performance in terms of NDCG@10 on a independent data set) of the mono-modal classifiers is kept for the test.

For the *MM* model, a set of parameters has to be learned for each class corresponding to the weights given to each centroid by the model. The class parameters are learned using a logistic regression where a positive target (+1) is associated to data points that belongs to the class and negative target (0) for unlabeled points. For classes with less than 30 data points, we use an uniform convex combination of the weights.

Note that the number of nearest-neighbors used for computing "cross-modal" centroids has been set to 10. The results obtained on 5, 20 and 30 nearest-neighbors lead to the same observations.

Table 3. Table presenting the results for a foldering task averaged on 5 chronological ordered training-test pairs for the Lokay's, Sanders's and Williams's mailboxes. The state-of-the art methods late fusion (*LF*) and early fusion (*EF*) are compared with the proposed multi-modal framework. The reported performance measure for this mono-label classification task are the Recall@1,@5, @10, the NDCG@10 and the NDCG. The hyper-parameters of each algorithm has been tuned on an independent validation set.

Alg.	R@1	R@3	R@5	R@10	NDCG@10	NDCG
Lokay's mailbox						
MM	0.80 +/- 0.04	0.92 +/- 0.05	0.95 +/- 0.04	0.96 +/- 0.04	0.89 +/- 0.04	0.89 +/- 0.04
LF	0.77 +/- 0.04	0.90 +/- 0.05	0.94 +/- 0.04	0.96 +/- 0.04	0.87 +/- 0.04	0.87 +/- 0.04
EF	0.81 +/- 0.05	0.91 +/- 0.06	0.95 +/- 0.05	0.96 +/- 0.04	0.89 +/- 0.05	0.89 +/- 0.05
Sanders's mailbox						
MM	**0.82** +/- 0.12	**0.86** +/- 0.13	**0.88** +/- 0.10	**0.92** +/- 0.12	**0.86** +/- 0.12	**0.87** +/- 0.11
LF	0.70 +/- 0.13	0.84 +/- 0.13	0.86 +/- 0.13	0.90 +/- 0.15	0.80 +/- 0.13	0.82 +/- 0.12
EF	0.78 +/- 0.14	0.82 +/- 0.14	0.86 +/- 0.16	0.89 +/- 0.15	0.83 +/- 0.14	0.84 +/- 0.13
Williams's mailbox						
MM	0.68 +/- 0.34	0.80 +/- 0.38	0.81 +/- 0.38	0.81 +/- 0.36	0.76 +/- 0.36	0.77 +/- 0.33
LF	0.74 +/- 0.37	0.80 +/- 0.39	0.81 +/- 0.38	0.81 +/- 0.38	0.78 +/- 0.38	0.79 +/- 0.35
EF	0.74 +/- 0.37	0.80 +/- 0.38	0.81 +/- 0.38	0.78 +/- 0.37	0.78 +/- 0.37	0.79 +/- 0.34

5.4 Results and Discussion

Foldering Task: The obtained results for the *MM*, *EF* and *LF* are reported in Table 2 and Table 3. The retrieval measure performance are the recall at rank 1 (R@1), the recall at rank 3 (R@3), the recall at rank 5 (R@5) and the recall at rank 10 (R@10), knowing that, for each data point, there is only one relevant folder. We measure also the normalized discounted cumulative gain, limited to rank 10 (NDCG@10) and on the whole set of folders (NDCG). The reported scores are the results averaged over the 5 sequential training-test pairs. Clearly, for 4 of the 7 mailboxes (namely: Beck, Farmer, Kaminski and Sanders) the proposed *MM* framework outperforms the *EF* and the *LF* strategies. The scores in bold mean that the performances are significantly better (verified by a signed test with a p-value < 0.05). Moreover, the *EF* and *LF* methods never outperforms the *MM* approach on all the measures simultaneously. Although the *LF* is generally the preferred approach for a foldering task on this data set (see, e.g. [6]), we observe in our tests that *EF* is always better or equivalent for all the performance measures than *LF*. Hence, we argue that often *EF* is not correctly used. Indeed, it is important to normalize independently each view before fusion due to the semantic difference that may exist between the views. The results reported on William's mailbox have a large variance. This is because, at the second time step, new folders have been created by the user for which no emails or a few were present in the training. In the real world, this realistic case often happen, therefore designing sequential training-test splits is critical in order to fairly assess the performance of the system.

Recipient Proposal. The results are reported for three different mailboxes in Table 4. The recipient proposal task is a multi-label classification task for which the performance are measured using the macroF1 (maF1), the mean average precision (MAP) and the NDCG. For the MM framework, we also report the

Table 4. Averaged performance measures on 7 different time slots of size 10 % (i.e. training size of 10 %) for Kaminski's, Dasovich's and Jone's mailboxes on a recipient proposal task. The state-of-the art methods late fusion (LF) and early fusion (EF) are compared with the proposed multi-modal framework. The reported performance measure for this multi-label classification task are the maF1, the MAP and the NDCG. The hyper-parameters of each algorithm has been tuned on an independent validation set.

Model	maF1	MAP	NDCG
Kaminski's mailbox			
$C^{(\text{text})}$	0.16 +/- 0.03	0.24 +/- 0.05	0.40 +/- 0.07
$C^{(\text{text})[\text{text}]}$	0.38 +/- 0.06	0.44 +/- 0.09	0.55 +/- 0.11
$C^{(\text{text})[\text{participant}]}$	0.11 +/- 0.02	0.13 +/- 0.03	0.24 +/- 0.04
$C^{(\text{participant})}$	0.04 +/- 0.01	0.11 +/- 0.01	0.26 +/- 0.04
$C^{(\text{participant})[\text{participant}]}$	0.11 +/- 0.01	0.21 +/- 0.02	0.37 +/- 0.04
$C^{(\text{participant})[\text{text}]}$	0.02 +/- 0.01	0.04 +/- 0.01	0.15 +/- 0.02
$\alpha C^{(\text{text})} + (1-\alpha)C^{(\text{participant})}$	0.18 +/- 0.03	0.26 +/- 0.05	0.42 +/- 0.07
EF	0.29 +/- 0.16	0.34 +/- 0.18	0.45 +/- 0.19
LF	0.30 +/- 0.16	0.35 +/- 0.18	0.46 +/- 0.19
RSV	**0.40** +/- 0.05	**0.46** +/- 0.08	**0.57** +/- 0.11
Dasovich's mailbox			
$C^{(\text{text})}$	0.61 +/- 0.14	0.25 +/- 0.06	0.46 +/- 0.04
$C^{(\text{text})[\text{text}]}$	0.66 +/- 0.14	0.28 +/- 0.06	0.48 +/- 0.07
$C^{(\text{text})[\text{participant}]}$	0.57 +/- 0.14	0.25 +/- 0.03	0.47 +/- 0.02
$C^{(\text{participant})}$	0.62 +/- 0.08	0.30 +/- 0.04	0.49 +/- 0.03
$C^{(\text{participant})[\text{participant}]}$	0.74 +/- 0.15	0.38 +/- 0.04	0.54 +/- 0.04
$C^{(\text{participant})[\text{text}]}$	0.78 +/- 0.09	0.37 +/- 0.04	0.49 +/- 0.03
$\alpha C^{(\text{text})} + (1-\alpha)C^{(\text{participant})}$	0.67 +/- 0.12	0.31 +/- 0.04	0.50 +/- 0.04
EF	0.36 +/- 0.06	0.41 +/- 0.06	0.52 +/- 0.06
LF	0.37 +/- 0.06	0.41 +/- 0.06	0.52 +/- 0.06
RSV	**0.79** +/- 0.07	**0.40**+/- 0.08	**0.55**+/- 0.06
Jone's mailbox			
$C^{(\text{text})}$	0.86 +/- 0.06	0.41 +/- 0.07	0.55 +/- 0.05
$C^{(\text{text})[\text{text}]}$	0.90 +/- 0.05	0.45 +/- 0.06	0.56 +/- 0.05
$C^{(\text{text})[\text{participant}]}$	0.83 +/- 0.07	0.39 +/- 0.06	0.54 +/- 0.04
$C^{(\text{participant})}$	0.80 +/- 0.05	0.40 +/- 0.07	0.55 +/- 0.05
$C^{(\text{participant})[\text{participant}]}$	0.86 +/- 0.07	0.47 +/- 0.06	0.60 +/- 0.04
$C^{(\text{participant})[\text{text}]}$	0.79 +/- 0.04	0.38 +/- 0.06	0.54 +/- 0.05
$\alpha C^{(\text{text})} + (1-\alpha)C^{(\text{participant})}$	0.86 +/- 0.04	0.41 +/- 0.06	0.55 +/- 0.04
EF	0.46 +/- 0.12	0.50 +/- 0.09	0.59 +/- 0.06
LF	0.46 +/- 0.12	0.12 +/- 0.09	0.59 +/- 0.06
RSV	**0.91** +/- 0.04	**0.50** +/- 0.06	**0.60** +/- 0.05

results obtained individually by each centroid. Moreover, in addition of reporting the results obtained by the *EF* and *LF* baselines, we also report the results obtained by the MM-baseline which consists of using only a combination of the mono-modal centroids.(in other words, the combination of the textual centroid and the social centroid : $\alpha C^{(\text{text})} + (1-\alpha)C^{(\text{participant})}$). On the three tested mailboxes, the proposed MM framework outperforms the *EF* and the *LF* and the simple combination of the textual and the social centroids. The scores in bold mean that the performances are significantly better (verified by a signed test with a p-value < 0.05). Notice that, on the Kaminski's mailbox, textual centroids obtained a better score than the social centroids, while on the Dasovitch's mailbox social centroids obtained a better score than the textual centroids.

6 Related Work

In a fully supervised setting, as pointed out in [3], multi-view learning usually performs worse than learning on the union of all views. Hence, a few has been done in this field. For instance, [12] proposed a method that combines a two stage learning (KCCA followed by SVM) into a single optimization termed "SVM-2K". Others have considered working on a graph representation of the data. For instance, [13] exploited hyperlinks between web pages in order to improve traditional classification tasks using only the content. [14] studied the composition of kernels in order to improve the performance of a soft-margin support vector machine classifier. [15, 16] used both local text in a document as well as the distribution of the estimated classes of other documents in its neighborhood, to refine the class distribution of the document being classified. Their framework has been tested for the semi-supervised and fully-supervised classification as well. More recently, [17] proposes to learn the weights of a namely "supervised random walk" using both the information from the network structure and the attribute data.

In this paper we introduce an extension of the classical Rocchio classification algorithm also known as the nearest centroid or nearest prototype classifier (see [18]) to the multi-modal case. An extension of this algorithm using kernel-based similarities has been introduced in [19]. A probabilistic variant of the Rocchio classifier has been proposed in [20].

7 Conclusions

In this work, we introduced a novel and simple algorithm in order to deal with multi-modal fully supervised classification. To this purpose, we extended the traditional Rocchio classification algorithm by defining mono-modal and multi-modal centroids. The introduced framework has the advantage of searching for a high consensus among views using scores reflecting the interactions between the different existing modes. We showed on two different tasks – a foldering task and recipient prediction task – that the proposed multi-modal framework outperforms state-of-the-art approaches: the early fusion and the late fusion. As

further research, we would like to investigate multi-view learning with latent spaces and interactions between latent variables.

References

[1] Abney, S.P.: Bootstrapping. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 360–367 (2002)

[2] Zhu, X.: Semi-supervised learning literature survey. Technical report (2008)

[3] Ruping, S., Scheffer, T.: Learning with multiple views proposal for an icml workshop. In: Proceedings of the ICML 2005 Workshop on Learning With Multiple Views, Bonn, Germany, August 11, pp. 1–7 (2005)

[4] Manning, C., Raghavan, P., Schutze, H.: Introduction to information retrieval. Cambridge University Press (2008)

[5] Bekkerman, R., McCallum, A., Huang, G.: Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora. Technical report, University of Massachusetts (2004)

[6] Tam, T., Ferreira, A., Lourenço, A.: Automatic Foldering of Email Messages:A Combination Approach. In: Baeza-Yates, R., de Vries, A.P., Zaragoza, H., Cambazoglu, B.B., Murdock, V., Lempel, R., Silvestri, F. (eds.) ECIR 2012. LNCS, vol. 7224, pp. 232–243. Springer, Heidelberg (2012)

[7] Liu, T., Xu, J., Qin, T., Xiong, W., Li, H.: Letor: Benchmark dataset for research on learning to rank for information retrieval. In: Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval, pp. 3–10 (2007)

[8] Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 391–398. ACM (2007)

[9] Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 271–278 (2007)

[10] Clinchant, S., Renders, J.-M., Csurka, G.: Trans-Media Pseudo-Relevance Feedback Methods in Multimedia Retrieval. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 569–576. Springer, Heidelberg (2008)

[11] Klimt, B., Yang, Y.: The Enron Corpus: A New Dataset for Email Classification Research. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 217–226. Springer, Heidelberg (2004)

[12] Farquhar, J.D.R., Hardoon, D.R., Meng, H., Shawe-Taylor, J., Szedmák, S.: Two view learning: Svm-2k, theory and practice. In: Proceedings of Advances in Neural Information Processing Systems, pp. 355–362 (2005)

[13] Slattery, S., Mitchell, T.: Discovering test set regularities in relational domains. In: Proceedings of the 7th International Conference on Machine Learning (ICML 2000), pp. 895–902 (2000)

[14] Joachims, T., Cristianini, N., Shawe-Taylor, J.: Composite kernels for hypertext categorisation. In: Proceedings of the International Conference on Machine Learning (ICML 2001), pp. 250–257 (2001)

[15] Chakrabarti, S., Dom, B., Indyk, P.: Enhanced hypertext categorization using hyperlinks. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 307–318 (1998)

[16] Oh, H., Myaeng, S., Lee, M.: A practical hypertext catergorization method using links and incrementally available class information. In: Proceedings of the 23rd International ACM Conference on Research and Development in Information Retrieval (SIGIR 2000), pp. 264–271. ACM (2000)
[17] Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, pp. 635–644 (2011)
[18] Tibshirani, R., Hastie, T., Narasimhan, B., Chu, G.: Diagnosis of multiple cancer types by shrunken centroids of gene expression. Proceedings of the National Academy of Sciences **99**(10) 99(10), 6567 (2002)
[19] Scholkopf, B., Smola, A.: Learning with kernels. The MIT Press (2002)
[20] Joachims, T.: A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In: Proceedings of International Conference on Machine Learning (ICML 1997), pp. 143–151 (1997)

Label-Noise Robust Logistic Regression and Its Applications

Jakramate Bootkrajang and Ata Kabán

School of Computer Science, The University of Birmingham,
Birmingham, B15 2TT, UK
{J.Bootkrajang,A.Kaban}@cs.bham.ac.uk

Abstract. The classical problem of learning a classifier relies on a set of labelled examples, without ever questioning the correctness of the provided label assignments. However, there is an increasing realisation that labelling errors are not uncommon in real situations. In this paper we consider a label-noise robust version of the logistic regression and multinomial logistic regression classifiers and develop the following contributions: (i) We derive efficient multiplicative updates to estimate the label flipping probabilities, and we give a proof of convergence for our algorithm. (ii) We develop a novel sparsity-promoting regularisation approach which allows us to tackle challenging high dimensional noisy settings. (iii) Finally, we throughly evaluate the performance of our approach in synthetic experiments and we demonstrate several real applications including gene expression analysis, class topology discovery and learning from crowdsourcing data.

1 Introduction

In the context of supervised learning, a classification rule is to be derived from a set of labelled examples. Regardless of the learning approach used, the induction of the classification rule crucially relies on the given class labels. Unfortunately, there is no guarantee that the class labels are all correct. The presence of class label noise inherent in training samples has been reported to deteriorate the performance of the existing classifiers in a broad range of classification problems [12,25,21]. Remarkably, examples of mislabelling have been reported even in biomedical sciences where the number of instances is only of the order of tens [1,15,26]. There is an increasing research literature that aims to address the issues related to learning from samples with noisy class label assignments. The seemingly straightforward approach is by means of data preprocessing where any suspect samples are removed or relabelled [3,2,9]. However, these approaches hold the risk of removing useful data too, especially when the number of training examples is limited.

In this paper, we take a model based approach and consider a label-noise robust logistic regression and multinomial logistic regression. There are already several works employing latent variable models of this kind, especially in the field of epidemiology, econometrics and computer-aided diagnosis (see [20,7,23]

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 143–158, 2012.

and references therein), and more recently for learning from crowds [23]. Our approach develops these ideas further while it differs in certain respects. [20] studied label-noise robust logistic regression with known label flipping probabilities but they reckon problems when these probabilities are unknown. In turn, we try to learn the classifier jointly with estimating the label flipping probabilities. The robust model discussed in [7] is also structurally similar to ours although they provided no algorithmic solution to learning the model. In contrast, one of our novel contributions in this paper is to develop an efficient learning algorithm together with a proof of its convergence. The recent work in [23] focuses on learning from multiple noisy labels and demonstrate that multiple sets of noisy labels increases performance. In contrast, our goal is to learn with a single set of noisy labels – which is considerably harder. In addition, we develop a novel sparsity-promoting regularisation approach which allows us to tackle challenging high dimensional noisy settings.

2 Label-Noise Robust Logistic Regression

We now describe the label-noise robust logistic regression (rLR) model. We will use the term 'robust' to differentiate this from traditional logistic regression (LR). Consider a set of training data $D = \{(\mathbf{x}_1, \tilde{y}_1), \ldots, (\mathbf{x}_N, \tilde{y}_N)\}$, where $\mathbf{x}_n \in \mathbb{R}^m$ and $\tilde{y}_n \in \{0, 1\}$, where $\tilde{y}_n$ denotes the observed (possibly noisy) label of $\mathbf{x}_n$. In the classical scenario for binary classification, the log likelihood is defined as:

$$\sum_{n=1}^{N} \tilde{y}_n \log p(\tilde{y} = 1|\mathbf{x}_n, \mathbf{w}) + (1 - \tilde{y}_n) \log p(\tilde{y} = 0|\mathbf{x}_n, \mathbf{w}). \tag{1}$$

where $\mathbf{w}$ is the weight vector orthogonal to the decision boundary and it determines the orientation of the separating plane. If all the labels were presumed to be correct, we would have $p(\tilde{y} = 1|\mathbf{x}_n, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}_n) = \frac{1}{1+e^{(-\mathbf{w}^T\mathbf{x}_n)}}$ and whenever this is above 0.5 we would decide that $\mathbf{x}_n$ belongs to class 1.

However, when label noise is present, making predictions in this way is no longer valid. Instead we will introduce a latent variable y, to represent the true label, and we model $p(\tilde{y} = k|\mathbf{x}_n, \mathbf{w})$ as the following:

$$p(\tilde{y} = k|\mathbf{x}_n, \mathbf{w}) = \sum_{j=0}^{1} p(\tilde{y} = k|y = j)p(y = j|\mathbf{x}_n, \mathbf{w}) \stackrel{def}{=} S_n^k \tag{2}$$

where $k \in \{0, 1\}$. Therefore, instead of Eq.(1), we define the log likelihood of our model as the following:

$$\mathcal{L}(\mathbf{w}) = \sum_{n=1}^{N} \tilde{y}_n \log S_n^1 + (1 - \tilde{y}_n) \log S_n^0 \tag{3}$$

In Eq.(2), $p(\tilde{y} = k|y = j) \stackrel{def}{=} \gamma_{jk}$ represents the probability that the label has flipped from the true label j into the observed label k. These parameters

form a transition table that we will refer to as the 'gamma matrix' from now on. Now, to classify a novel data point $\mathbf{x}_q$, we predict that $\hat{y}_q = 1$ whenever $p(y = 1|\mathbf{x}_q, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}_q) = \frac{1}{1+e^{(-\mathbf{w}^T\mathbf{x}_q)}}$ returns a value greater than 0.5, and $\hat{y}_q = 0$ otherwise.

2.1 Parameter Estimation with Multiplicative Updates

Learning the rLR requires us to estimate the weight vector $\mathbf{w}$ as well as the label-flipping probabilities γ_{jk}. To optimise the weight vector, we can use any nonlinear optimiser. Here we decided to employ conjugate gradients because of its well known computational efficiency, which basically performs the Newton update step along the direction $\mathbf{u} = \mathbf{g} - \mathbf{u}^{old}\beta$, where $\mathbf{g} = \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w})$ is the gradient:

$$\mathbf{g} = \sum_{n=1}^{N}\left[\left(\frac{\tilde{y}_n(\gamma_{11}-\gamma_{01})}{S_n^1} + \frac{(1-\tilde{y}_n)(\gamma_{10}-\gamma_{00})}{S_n^0}\right)\sigma(\mathbf{w}^T\mathbf{x}_n)(1-\sigma(\mathbf{w}^T\mathbf{x}_n))\cdot\mathbf{x}_n\right] \quad (4)$$

One may verify that setting γ_{01} and γ_{10} to 0 and γ_{00}, γ_{11} to 1, after some algebra, Eq.(4) will reduce to the well-known gradient expression of classical logistic regression. The parameter β that works best in practice can be obtained from the Hestenes-Stiefel formula, $\beta = \frac{\mathbf{g}^T(\mathbf{g}-\mathbf{g}^{old})}{(\mathbf{u}^{old})^T(\mathbf{g}-\mathbf{g}^{old})}$. Then, the update equation for $\mathbf{w}$ is simply the following:

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \frac{\mathbf{g}^T\mathbf{u}}{\mathbf{u}^T\mathbf{H}\mathbf{u}}\mathbf{u}, \quad (5)$$

where $\mathbf{H}$ is the Hessian matrix.

To obtain the updates for the label-flipping probabilities, we introduce Lagrange multipliers to ensure that $\gamma_{00} + \gamma_{01} = 1$ and $\gamma_{10} + \gamma_{11} = 1$. Conveniently, after some algebra, the stationary equations yield the following multiplicative update equations:

$$\gamma_{00} = \frac{\gamma_{00}\sum_{n=1}^{N}\left[\frac{(1-\tilde{y}_n)}{S_n^0}(1-\sigma(\mathbf{w}^T\mathbf{x}_n))\right]}{\gamma_{00}\sum_{n=1}^{N}\left[\frac{(1-\tilde{y}_n)}{S_n^0}(1-\sigma(\mathbf{w}^T\mathbf{x}_n))\right] + \gamma_{01}\sum_{n=1}^{N}\left[\frac{\tilde{y}_n}{S_n^1}(1-\sigma(\mathbf{w}^T\mathbf{x}_n))\right]} \quad (6)$$

$$\gamma_{11} = \frac{\gamma_{11}\sum_{n=1}^{N}\left[\frac{\tilde{y}_n}{S_n^1}\sigma(\mathbf{w}^T\mathbf{x}_n)\right]}{\gamma_{10}\sum_{n=1}^{N}\left[\frac{(1-\tilde{y}_n)}{S_n^0}\sigma(\mathbf{w}^T\mathbf{x}_n)\right] + \gamma_{11}\sum_{n=1}^{N}\left[\frac{\tilde{y}_n}{S_n^1}\sigma(\mathbf{w}^T\mathbf{x}_n)\right]} \quad (7)$$

Our rLR training algorithm is then to alternate between updating each parameter in turn, until convergence. It is worth noting that the objective we are trying to optimise is non-convex. Hence, the result will inevitably depend on the initialisation of those parameters, and we will return to this point in the experimental section. However, convergence to a local optimum is guaranteed, as we shall see shortly.

2.2 Multiclass Label-Noise Robust Logistic Regression

It is both useful and straightforward to generalise the two-class rLR of the previous section to multiclass problems. We again introduce the true class label y as a latent variable and write:

$$p(\tilde{y}=k|\mathbf{x}_n,\mathbf{w}_k)=\sum_{j=0}^{K-1}p(\tilde{y}=k|y=j)\cdot p(y=j|\mathbf{x}_n,\mathbf{w}_j)\stackrel{def}{=}S_n^k \tag{8}$$

where $p(y=k|\mathbf{x}_n,\mathbf{w}_k)$ is modelled using a softmax function, $\frac{e^{(\mathbf{w}_k^T\mathbf{x}_n)}}{\sum_{l=0}^{K-1}e^{(\mathbf{w}_l^T\mathbf{x}_n)}}$, and $\mathbf{w}_k$ is the weight vector corresponding to class k. The maximum likelihood (ML) estimate of $\mathbf{w}_k$ is obtained by maximising the data log-likelihood,

$$\mathcal{L}(\mathbf{w})=\sum_{n=1}^{N}\sum_{k=0}^{K-1}\delta(\tilde{y}_n=k)\log S_n^k \tag{9}$$

where $\delta(\tilde{y}_n=k)$ is the Kronecker delta function that gives the value 1 when its argument is true and the value 0 otherwise. The optimisation is again accomplished using the conjugate gradient method where the gradient becomes:

$$\mathbf{g}=\sum_{n=1}^{N}\sum_{k=0}^{K-1}\frac{\delta(\tilde{y}_n=k)}{S_n^k}\times\frac{e^{(\mathbf{w}_c^T\mathbf{x}_n)}\mathbf{x}_n\left(\sum_{j=0}^{K-1}(\gamma_{ck}-\gamma_{jk})\cdot e^{(\mathbf{w}_j^T\mathbf{x}_n)}\right)}{\left(\sum_{l=0}^{K-1}e^{(\mathbf{w}_l^T\mathbf{x}_n)}\right)^2} \tag{10}$$

Further, the estimates of γ_{jk} in the gamma matrix again can be obtained by efficient multiplicative update equations:

$$\gamma_{jk}=\frac{1}{C}\times\gamma_{jk}\sum_{n=1}^{N}\frac{\delta(\tilde{y}_n=k)}{S_n^k}\cdot\frac{e^{(\mathbf{w}_j^T\mathbf{x}_n)}}{\sum_{l=0}^{K-1}e^{(\mathbf{w}_l^T\mathbf{x}_n)}}, \tag{11}$$

where the constant term C equals $\sum_{k=0}^{K-1}\gamma_{jk}\sum_{n=1}^{N}\frac{\delta(\tilde{y}_n=k)}{S_n^k}\times\frac{e^{(\mathbf{w}_j^T\mathbf{x}_n)}}{\sum_{l=0}^{K-1}e^{(\mathbf{w}_l^T\mathbf{x}_n)}}$.

To classify a new point, we decide $\hat{y}_q=\arg\max_k\frac{e^{(\mathbf{w}_k^T\mathbf{x}_q)}}{\sum_{l=0}^{K-1}e^{(\mathbf{w}_l^T\mathbf{x}_q)}}$.

3 Convergence of the Algorithm

We shall now prove that the learning algorithms proposed in the previous sections, for both rLR and rmLR, converge. The idea of the proof is to show that the objective function being optimised, Eq.(9) is nondecreasing under any of our parameter updates. Indeed, the maximisation w.r.t. the weight vector $\mathbf{w}$ by the conjugate gradient method (CG) enjoys the known property of CG to provide monotonically improving estimation of the target [8], which guarantees that an objective function being maximised is nondecreasing. Now, it remains to prove that our multiplicative updates for γ_{jk} are also guaranteed to be nondecreasing. To do this, we use the notion of an auxiliary function, in a similar spirit to the proofs in [14].

Definition 1. *$G(h, h')$ is an auxiliary function for $F(h)$ if*

$$G(h, h') \leq F(h), G(h, h) = F(h) \tag{12}$$

are satisfied.

The definition is useful because of the following lemma.

Lemma 1. *[14] If G is an auxiliary function, then F is nondecreasing under the update*

$$h^{i+1} = \arg\max_{h} G(h, h^i) \tag{13}$$

Proof. $F(h^{i+1}) \geq G(h^{i+i}, h^i) \geq G(h^i, h^i) = F(h^i)$

We will show that by defining an appropriate auxiliary function to the objective function Eq. (9), regarded as a function of Γ, the update equations Eq.(11) for γ_{jk} are guaranteed to converge to a local optimum.

Lemma 2. *Define*

$$G(\Gamma, \tilde{\Gamma}) = \sum_{n=1}^{N} \sum_{k=0}^{K-1} \delta(\tilde{y}_n = k) \sum_{j=0}^{K-1} \frac{\tilde{\gamma}_{jk} p(y = j|\boldsymbol{x}_n, \boldsymbol{w})}{\sum_{l=0}^{K-1} \tilde{\gamma}_{lk} p(y = l|\boldsymbol{x}_n, \boldsymbol{w})} \times \left(\log \tilde{\gamma}_{jk} p(y = j|\boldsymbol{x}_n, \boldsymbol{w}) - \log \frac{\tilde{\gamma}_{jk} p(y = j|\boldsymbol{x}_n, \boldsymbol{w})}{\sum_{l=0}^{K-1} \tilde{\gamma}_{lk} p(y = l|\boldsymbol{x}_n, \boldsymbol{w})} \right) \tag{14}$$

This is an auxiliary function for

$$\mathcal{L}(\Gamma) = \sum_{n=1}^{N} \sum_{k=0}^{K-1} \delta(\tilde{y}_n = k) \log \sum_{j=0}^{K-1} \gamma_{jk} p(y = j|\boldsymbol{x}_n, \boldsymbol{w}) \tag{15}$$

Proof. For $G(\Gamma, \tilde{\Gamma})$ to be an auxiliary function it needs to satisfy the aforementioned conditions. It is straightforward to verify that $G(\Gamma, \Gamma) = \mathcal{L}(\Gamma)$. To show that $G(\Gamma^{i+1}, \Gamma^i) \leq \mathcal{L}(\Gamma^{i+1})$, we observe that:

$$\log \sum_{j=0}^{K-1} \gamma_{jk} p(y = j|\mathbf{x}_n, \mathbf{w}) \geq \sum_{j=0}^{K-1} \alpha_{jk} \log \left(\frac{\gamma_{jk} p(y = j|\mathbf{x}_n, \mathbf{w})}{\alpha_{jk}} \right), \tag{16}$$

by Jensen's inequality and due to the convexity of the log function. This inequality is valid for all non-negative α_{jk} that sum to one. Setting

$$\alpha_{jk} = \frac{\tilde{\gamma}_{jk} p(y = j|\mathbf{x}_n, \mathbf{w})}{\sum_{l=0}^{K-1} \tilde{\gamma}_{lk} p(y = l|\mathbf{x}_n, \mathbf{w})}, \tag{17}$$

we see that our objective function $\mathcal{L}(\Gamma)$ is always greater than or equal to the auxiliary function (14).

Lemma 3. *The multiplicative update rule of the label flipping probability* γ_{jk} *given in Eq. (11) is guaranteed to converge.*

Proof. The maximum of $G(\Gamma, \tilde{\Gamma})$ with respect to γ_{jk} is found by setting the derivative to zero:

$$\frac{dG(\Gamma, \Gamma^i)}{d\gamma_{jk}} = \sum_{n=1}^{N} \delta(\tilde{y}_n = k)\frac{\alpha_{jk}}{\gamma_j k} - \lambda = 0, \tag{18}$$

Using the fact that $\sum_j \gamma_{jk} = 1$, we obtain the value of the Lagrange multiplier λ. Putting it back into Eq. (18) we arrive at:

$$\gamma_{jk} = \frac{1}{C} \times \tilde{\gamma}_{jk} \sum_{n=1}^{N} \delta(\tilde{y}_n = k) \cdot \frac{p(y = j|\mathbf{x}_n, \mathbf{w})}{\sum_{l=0}^{K-1} \tilde{\gamma}_{lk} p(y = l|\mathbf{x}_n, \mathbf{w})}, \tag{19}$$

where C equals $\sum_{k=0}^{K-1} \tilde{\gamma}_{jk} \sum_{n=1}^{N} \delta(\tilde{y}_n = k) \frac{p(y=j|\mathbf{x}_n, \mathbf{w})}{\sum_{l=0}^{K-1} \tilde{\gamma}_{lk} p(y=l|\mathbf{x}_n, \mathbf{w})}$. Writing out posterior probability $p(y = j|\mathbf{x}_n, \mathbf{w})$ as a softmax function and noting that by definition $\sum_{l=0}^{K-1} \tilde{\gamma}_{lk} p(y = l|\mathbf{x}_n, \mathbf{w})$ equals S_n^k, Eq. (19) then takes the same form as the update rule in Eq. (11). Since $G(\Gamma, \tilde{\Gamma})$ is an auxiliary function, it is guaranteed that the value of $\mathcal{L}$ is nondecreasing under this update.

Theorem 1. *By alternating between the updates of the weight vector* $\boldsymbol{w}$ *while the matrix* Γ *is held fixed, and the updates of the elements of* Γ *while* $\boldsymbol{w}$ *is fixed, the objective function of rmLR is nondecreasing and is thus guaranteed to converge.*

Proof. The proof follows directly from the fact that optimising $\mathbf{w}$ using CG is monotonically nondecreasing and from *Lemma 3*, that optimising Γ is also nondecreasing. Consequently, the objective function being optimised is monotonically increasing under alternating these iterations.

Finally, note that as rmLR is a direct generalisation of rLR, the proof also covers the case of rLR.

3.1 Comparison with EM Based Optimisation

The algorithm developed in [23] in the context of multiple sets of noisy labels could also be instantiated for our problem, as an alternative to the above approach. The method in [23] proposes an EM algorithm where the true labels are the hidden variables. Instead, we had these hidden variable integrated out when optimising the parameters. It is hence interesting to see how they compare.

Similar to [23], let y_n be the hidden true labels, and denote $P_n = p(y_n = 1|\mathbf{x}, \mathbf{w}, \tilde{y}_n)$ the posterior of these. Then, the expected complete log likelihood (so-called Q-function) can then be written as:

$$\mathcal{Q}(\mathbf{w}, \Gamma) = \sum_{n=1}^{N} P_n \log(\gamma_{11}^{\tilde{y}_n} \gamma_{10}^{1-\tilde{y}_n} \sigma(\mathbf{w}^T \mathbf{x}_n)) + (1 - P_n) \log(\gamma_{01}^{\tilde{y}_n} \gamma_{00}^{1-\tilde{y}_n} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n))) \tag{20}$$

– The E-step involves optimising P_n based on given data and current estimated of γ_{jk}:

$$P_n = \frac{\gamma_{11}^{\tilde{y}_n}\gamma_{10}^{1-\tilde{y}_n}\sigma(\mathbf{w}^T\mathbf{x}_n)}{\gamma_{11}^{\tilde{y}_n}\gamma_{10}^{1-\tilde{y}_n}\sigma(\mathbf{w}^T\mathbf{x}_n) + \gamma_{01}^{\tilde{y}_n}\gamma_{00}^{1-\tilde{y}_n}(1-\sigma(\mathbf{w}^T\mathbf{x}_n))} \tag{21}$$

– The M-step then re-estimate the value of γ_{jk} using P_n from the E-step. For example γ_{11} can be update using:

$$\gamma_{11} = p(\tilde{y}_n = 1|y_n = 1) = \frac{\sum_{n=1}^{N} P_n\tilde{y}_n}{\sum_{n=1}^{N} P_n} \tag{22}$$

Now, observe that substituting the r.h.s. of P_n into the M-step equations, we recover our multiplicative form of updates – with one subtle but important difference: In the EM approach P_n is computed with old values of the parameters (from the previous iteration). Instead, our multiplicative updates use the latest fresh values of all the parameters they depend on. This implies that our algorithm has a better chance to converge in fewer iterations, and in addition it saves the storage cost of the posteriors P_n during the iterations. Worth noting also that P_n can be useful for interpretation — however we can compute this after convergence using the final values of the parameters.

4 Sparse Extension via a Bayesian-Regularised Generalised Lasso

In many real world problems, especially in biomedical domains, we are faced with high dimensional data with more features than observations, while only a subset of the features is relevant to the target. Sparsity-inducing regularisation approaches have been effective in such cases [19,24,4]. In this section we show that our model can be extended to support such regularised estimation. Akin to generalised Lasso [24], we will employ L1-regularisation terms on each component of $\mathbf{w}$. We should mention that other approaches such as Automatic Relevance Determination based on t-prior [19] could also be used in a similar manner.

Our regularised objective is now the following:

$$\max_{\mathbf{w}} \sum_{n=1}^{N} \log p(\tilde{y}_n|\mathbf{x}_n, \mathbf{w}) - \sum_{i=1}^{m} \alpha_i|w_i| \tag{23}$$

where m is the number of features and α_i are Lagrange multipliers that balance between fitting the data well and having small parameter values. Eq.(23) is not differentiable at the origin. To counter this, here we adopt a very simple, yet effective smooth approximation originally proposed in [16] for Lq-regularisation. This is to approximate $|w_i| \approx (w_i^2 + \gamma)^{1/2}$, and we have set $\gamma = 10^{-8}$ in the reported experiments.

Now, the regularisation parameters α_i need to be determined. The common approach would be to use cross-validation — however, this would need to make use of the labels of the validation sets, which have no guarantee of being correct

in our problem setting. We turn to a Bayesian regularisation approach where α_i is eliminated from the model by marginalisation. Bayesian regularisation was found comparable in performance to cross-validation [5], and in particular it was also demonstrated to be effective for L1-regularised logistic regression [4].

Our version will be different from the one in [4] mainly because the latter is tied to their specific implementation in that $\alpha_i = \alpha$ for all i for which $w_i \neq 0$, and a Jeffreys hyperprior is posited only on these non-zero components. This requires an estimate of the number of non-zeros. Instead, we will simply posit independent Jeffreys priors on each α_i and let the ones that are not supported by the data die out naturally.

We begin by considering a Bayesian interpretation of the problem in Eq.(23). That is, the posterior distribution of $\mathbf{w}$, conditional on $\boldsymbol{\alpha}$, can be written as

$$p(\mathbf{w}|\mathcal{D}, \boldsymbol{\alpha}) \propto p(\mathcal{D}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha}). \tag{24}$$

Now the first term on the r.h.s is the data likelihood, while the second term corresponds to our regularisation term. If we take logarithm of the whole expression, we have: $\log p(\mathbf{w}|\mathcal{D}, \boldsymbol{\alpha}) = \log p(\mathcal{D}|\mathbf{w}) + \log p(\mathbf{w}|\boldsymbol{\alpha}) + const.$

Thus, the regularisation term in Eq.(23) is just the negative logarithm of the conditional prior distribution, conditioned on $\boldsymbol{\alpha}$, up to an additive constant. The conditional prior $p(\mathbf{w}|\boldsymbol{\alpha})$ is then given by a product of independent Laplace distributions with parameters $\boldsymbol{\alpha}$: $p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=1}^{m} p(w_i|\alpha_i) = \frac{\prod_{i=1}^{m}\alpha_i}{2^m} \exp\left(-\sum_{i=1}^{m}\alpha_i|w_i|\right) \approx \frac{\prod_{i=1}^{m}\alpha_i}{2^m} \exp\left(-\sum_{i=1}^{m}\alpha_i(w_i^2+\gamma)^{1/2}\right)$. Now, we want to eliminate its dependency on $\boldsymbol{\alpha}$ by marginalisation, i.e. to have the marginal prior as the following:

$$p(\mathbf{w}) = \int p(\mathbf{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})\mathrm{d}\boldsymbol{\alpha} \tag{25}$$

For this, we posit Jeffrey's priors, $p(\alpha_i) \propto \frac{1}{\alpha_i}$, on each α_i. This is the non-informative improper limit of a Gamma prior, and it has the advantage that it is parameter-free. Substituting this and $p(w_i|\alpha_i)$ and performing the integral in Eq. (25), $\int_0^\infty \frac{1}{\alpha_i}\frac{\alpha_i}{2}\exp(-\alpha_i(w_i^2+\gamma)^{1/2})d\alpha_i = \frac{1}{2(w_i^2+\gamma)^{1/2}}$,we obtain the following marginal prior:

$$p(\mathbf{w}) = \frac{1}{2}\prod_{i=1}^{m}\frac{1}{(w_i^2+\gamma)^{1/2}}, \tag{26}$$

which implies that negative log of the marginal prior $-\log p(\mathbf{w}) = \sum_{i=1}^{m}\log\left((w_i^2+\gamma)^{1/2}\right) + const.$ Now, replacing the regularisation term that appears in Eq.(23) by the above marginal prior, and taking derivative with respect to the model parameters w_i again as we did before, now we have:

$$\frac{\partial\mathcal{L}(\mathbf{w})}{\partial w_i} = \frac{\partial}{\partial w_i}\sum_{n=1}^{N}\left[\tilde{y}\log(S_n^1) + (1-\tilde{y})\log(S_n^0)\right] + \frac{1}{(w_i^2+\gamma)^{1/2}}\frac{\partial}{\partial w_i}\left(\sum_{i=1}^{m}\log((w_i^2+\gamma)^{1/2})\right) \tag{27}$$

From this, we read off the estimates of the regularisation parameters as:

$$\alpha_i = \frac{1}{(w_i^2 + \gamma)^{1/2}} \tag{28}$$

The optimisation of the log-likelihood is then to alternate between optimising $\mathbf{w}$ along with updating α_i according to Eq.(28) until convergence in reached, and of course, we alternate this with the fixed point update equations of the label flipping probabilities given in the previous sections. Generalising the sparse regression procedure described in this section to multi-class settings is straightforward.

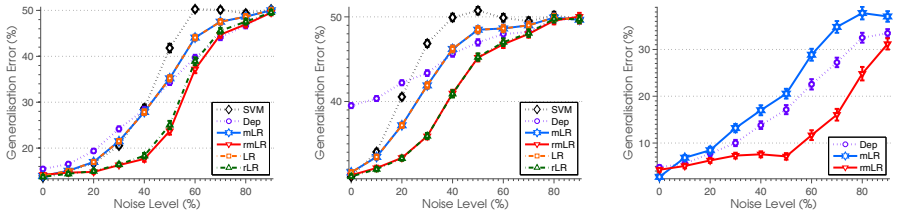
5 Experimental Validation and Applications

5.1 Simulated Label Noise

Before presenting real applications where no ground truth is available for an objective validation, we first assess our algorithm on real world data sets using artificial class label noise. We used three standard data sets from the UCI repository: *Boston*, *Liver* (binary) and *Iris* (multiclass). Since it has been shown theoretically that symmetric label noise is relatively harmless, for example see [18], here only asymmetric label noise of various levels was artificially injected for the purpose of systematic testing. In addition, we will compare our result to two existing methods: (i) Depuration [2], which is a non-parametric method based on nearest neighbours, previously proposed for the same problem of dealing with label-noise in classification; and (ii) Support Vector Machines (SVM), which has the well-known margin and slack-variable mechanism built in, and which may provide some robustness. The reason to compare with SVM is to find out to what extent class label noise could be considered to be a normal part of any classification problem — and conversely, to what extent it actually needs the special treatment that we developed in the previous sections. Code that reproduces the results of our experiments is available on request.

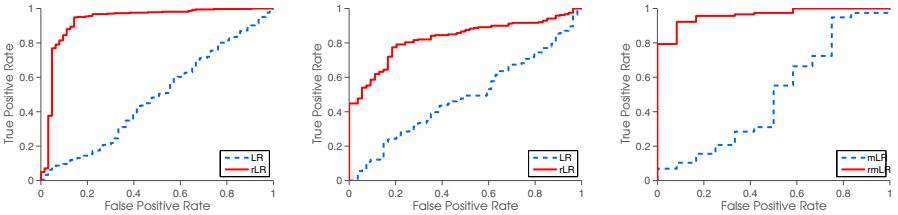
It should be noted that when applying Depuration and SVM, we again face with the problem of model selection. A general approach to model selection is a standard cross validation technique. Although this works well in a traditional setting where all class labels are correct, it is no longer applicable here. This problem was also reported in [13], However, the solution they resort to is simply to assume that there is a trusted validation set available. This may be unrealistic in many real situations, and especially so in small-sample problems as in [26].

Figure 1 summarises our results on three classification data sets. It is clear that both rLR and rmLR outperform each algorithm on each of the data sets tested. Depuration (denoted as 'Dep' in the figure) tends to perform well in a very high level of noise (i.e. 50% upwards) while at the lower range, its performance is slightly worse. The comparative results with SVM also demonstrate convincingly that class label noise does need special attention and it is naive to consider label noise as a normal part of classification problems. We see that our algorithm developed explicitly for this problem does indeed achieve improved classification performance overall.



(a) Boston (Binary) (b) Liver (Binary) (c) Iris (Multiclass)

Fig. 1. Classification errors on real world data sets when the labels are artificially flipped asymmetrically



(a) Boston (Binary) (b) Liver (Binary) (c) Iris (Multiclass)

Fig. 2. ROC curves. Labels are asymmetrically flipped at 30% noise.

Next, we assess our methods' ability to detect the instances that were wrongly labelled. There are two types of possible errors: (i) a false positive is when a point is believed to be mislabelled when in fact it is labelled correctly; and (ii) a false negative is when a point is believed to be labelled correctly when in fact its label is incorrect. A good way to summarise both, while also using the probabilistic output given by the sigmoid or the softmax functions, may be obtained by constructing the Receiver Operating Characteristic (ROC) curves. Figure 2 shows the ROC curves for all real world data sets tested, at a asymmetric noise level of 30%. Superimposed for reference we also plotted the ROC curves that correspond to the traditional classifier that believes that all points have the correct labels. The gap between the two curves is well apparent in all four cases tested, and it quantifies the gain obtained by our modelling approach in each setting. The area under the ROC curve signifies the probability that a randomly drawn and mislabelled example would be flagged by our method. For the sake of clarity of the graph, the results from Depuration and SVM were not included here as we have already seen that they are inferior to rLR and rmLR. We now turn to demonstrating real applications in several domains.

5.2 Application to Finding Mislabelled Gene Arrays in Colon Cancer Data

So far we presented controlled experiments where the label-noise was artificially created. It is now most interesting to demonstrate the effectiveness of our

approach on a data set whose labels are genuinely inaccurate. In this section we take the Colon Cancer data set [1]. This contains expression levels of 2000 genes from 40 tumour and 22 normal colon tissues, and there is some evidence in the biological literature that label noise may be present [6,15,11,21,26].

We split the data into 52 training points and 10 test points. In order to get a more reliable accuracy figure, we have excluded from the test set all of the instances which are suspected to be wrong based on the existing evidence. Instead, these instances will be placed in the training set in all the training-testing splits that we consider. We note that the number of instances affected by label noise is unequal for the two classes: approximately 20% of normal tissues were labelled as tumour while 10% of tumour tissues were labelled as normal. The nature of mislabelling corresponds to a slightly asymmetric flipping scenario that we have previously discussed. Hence the label noise will likely perturb the learning of traditional classifiers.

For illustrative purpose, we evaluate predictive performance on the test set averaged over 1000 training-testing splits and observe that rLR significantly outperform its traditional competitors and achieves an impressive accuracy with the error rate of 2.08 ± 0.055, while the performance of LR (3.66 ± 0.064) and SVM (4.08 ± 0.063) lag behind. We should note, these results are not directly comparable to other studies where the mislabelled points have not been excluded from the test set.

More importantly, biologists are interested in understanding the nature of data rather than classification accuracy figures. Here we demonstrate the use of algorithm for detecting the wrongly labelled instances. This is particularly useful in scientific applications, where a sample detected as potentially mislabelled could then be handed over to the domain expert for confirmation or further study. In Table 1 we compare the results of previous attempts at this problem with rLR.

The penultimate line gives the frequency rates of mislabelling detections computed from 20 independent runs on the whole data set, with independent random initialisation, and using $\sum_{j=0,j\neq\tilde{y}_n}^{K-1} p(y=j|\mathbf{x}_n,\mathbf{w})$ thresholded at 0.5 each time. Since the objective function that we optimise is non-convex, as mentioned previously, our greedy iterative algorithm finds one of its local optima at each run, and hence different runs can come up with different detections depending on the initialisation. A frequency rate of 1 means that the probability that the true label differs from the observed one for that point was estimated to be higher than 0.5 in all of the 20 repeated runs. A value of 0.15 means that it was estimated to be higher than 0.5 in 3 out of the 20 runs.

We observe that rLR can identify up to 8 distinct mislabelled points, and these cover all except one of the union of all previously identified mislabellings, and do not detect any other point outside this union. We also note that this total of 8 points were not identified at any single run of our algorithm either. This suggests that in this case having several local optima is not necessarily a bad thing for the application, as it allows us to have different views at the problem which may be more comprehensive than a simplified single view. The last row provides

Table 1. Identifying mislabelled points from the Colon Cancer data set. The first row is the 'gold standard' with biological evidence. The last two rows present our results (see text for explanations). The rest are the results from previous studies.

Source	Suspects identified									Extra samples identified
Alon et al. [1]	T2	T30	T33	T36	T37	N8	N12	N34	N36	
Furey et al. [6]		○	○	○		○		○	○	
Li et al. [15]		○	○	○				○	○	
Kadota et al. [11]	○				○	○		○	○	T6,N2
Malossini et al. [21]	○	○	○	○			○	○	○	T8,N2,N28,N29
reg-rLR by frequency	0.15	1.00	1.00	1.00	0	0.25	0.1	1.00	1.00	
reg-rLR degree of belief	0	0.70	0.66	0.76	0	0.54	0.54	0.59	0.60	

the degree of belief, ie. the actual probabilities from $\sum_{j=0,j\neq\tilde{y}_n}^{K-1} p(y=j|\mathbf{x}_n,\mathbf{w})$, without any thresholding, for the single best run out of the 20 independent trials, selected by the best minimum of the objective function being minimised by our algorithm. Seven points, namely T30,T33,T36,N8,N12,N34 and N36 were detected in this best run. The probabilities that we see here mean the confidence of each of these detections. Relating these back to the previous row of the table, the frequency rates, we see that those samples that were identified with a higher degree of belief (T33, T36, N34 and N36) also have a higher frequency of being detected.

5.3 Application to Structure Discovery: Inferring a Class-Topology in Multi-class Problems

The next experiment demonstrates a different use of our label-noise robust classifier, namely to infer the internal topological structure of the data classes. For many real-world classification tasks the labelling process is somewhat subjective as there is no clear-cut boundary between the classes. For example, in the case of classifying text messages according to topic, some instances could be assigned to more than one category. Thus, interpreting the gamma matrix as the adjacency matrix of a directed graph could reveal the internal structure of the data set under study. To demonstrate this idea, we employed rmLR on *Newsgroups*[1] data set. The corpus was subject to tokenisation, stop words removal, and Porter stemming to remove the word endings prior to cosine normalisation.

Figure 3 shows the graph derived from the gamma matrix as obtained from *10 Newsgroups*. Each node corresponds to a topic class while the length of an edge connecting two nodes represents the strength of relationship between them. The direction of arrows then correspond to the label flipping directions. It can be seen from this graph that 'atheism' and 'religion' are related topics by looking at the distance between the two as well as the bi-directional flipping relation,

[1] Originally the *Newsgroups* corpus comprises 20 classes of postings, We use the subset of 10 classes from [10], with term frequency count based encoding.

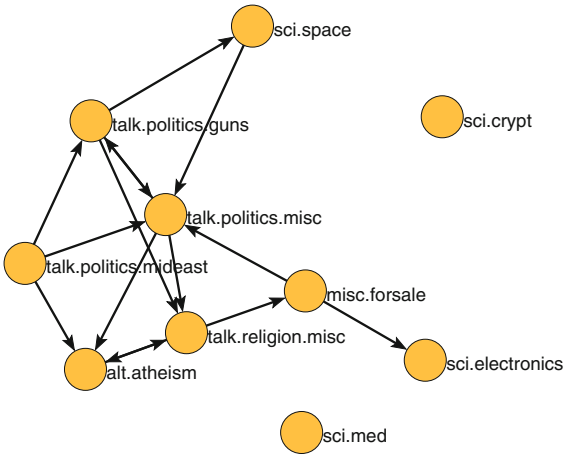


Fig. 3. Adjacency graph of the ten topics on the Newsgroups data set

which indeed agrees with our commonsense. Similar observation can also be made between the 'electronics' and 'for-sale' postings. Further, the graph also visually suggests various sub-groupings: for example, all classes related to politics are clustered nearer to each other.

5.4 Application to Learning from Crowds: Learning to Classify Images Using Cheaply Obtained Labelled Data

It is well reckoned that careful labelling of large amounts of data by human experts is extremely tiresome. Suppose we were to train a classifier to distinguish an image of 'bike' from other type of images. The standard machine learning approach is to collect training images and manually label each of them — rather labourious. Here, we suggest that we could reduce human expert intervention and obtain the training data cheaply using annotated data from search engines. By searching for images using keyword 'bike' we obtain a set of images that are loosely categorised into 'bike' class, and similarly 'not bike' class by using its negation. This allows us to acquire a large number of training data quickly and cheaply. The problem is of course that the annotations returned by the search engine are somewhat unreliable. This is where rLR comes into play. Here we collected 520 images using the keyword 'bike' and 520 images using the keyword 'not bike' that we call the *WebSearch*[2] dataset. We also manually label all images: a 'bike' image is one that contains a bike as its main object and we make no distinction between a bicycle and a motorbike, everything else is labelled as 'not bike'. This reveals 83 flips from 'bike' to 'not bike' images and 100 flips from 'not bike' to 'bike' category. The manually labelled set is only used for testing purposes. The images are passed through a series of preprocessing including

[2] Collected using Google image search engine: available upon request.

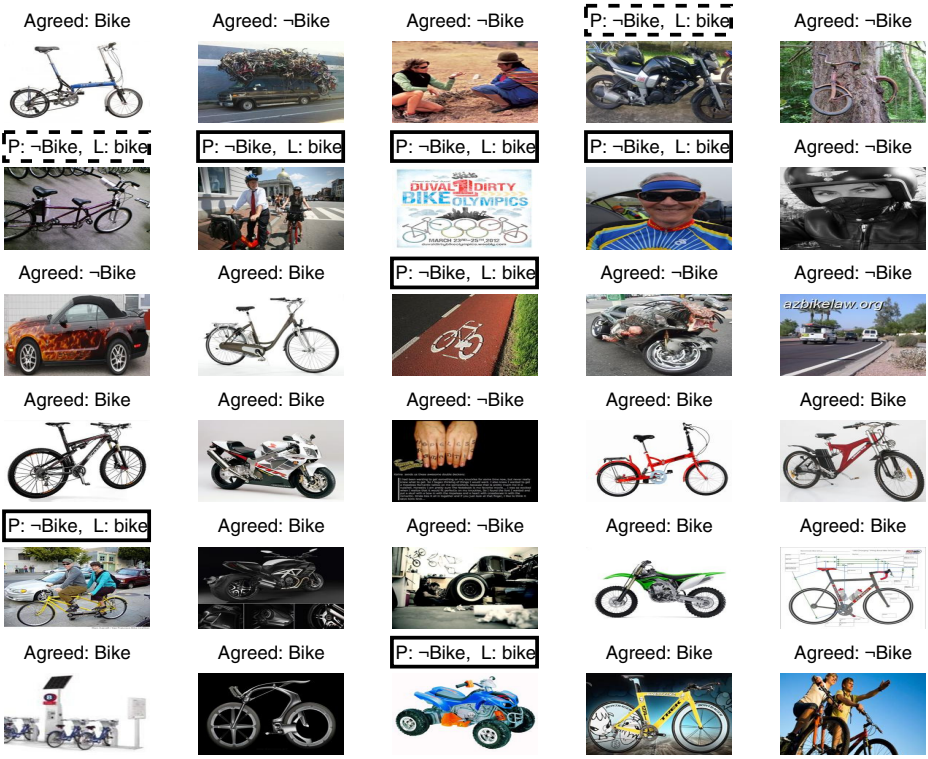


Fig. 4. Bike search result. P is the prediction from the classifier while L is the given label from search engine. Boxed instances are the ones that P and L don't agree while dotted boxes are false alarms.

extracting meaningful visual vocabulary using SIFT [17] and extracting texture information using LBP [22], which are ultimately transformed into a 10038-dimensional vector representation.

In Figure 4 we show examples of detecting mislabelled images. The top 30 test images sorted by their posterior probabilities are shown. We see that out of a total of 8 suspicious detections made (boxed), only 2 were false alarms (denoted by dotted box in the figure). Comparatively, the traditional LR model produced 4 false alarms (not shown). To give statistical figures, we then tested these two classifiers that were both trained on 90% of whole dataset using the cheap noisy labels from the search engine, and tested on the ramaining 10%, against the manual labels. We performed 100 independent bootstrap repetitions of this experiment. The average generalisation errors and standard errors were 15.67% $\pm$ 0.04 for rLR and 18.09% $\pm$ 0.04 for standard LR. The improvement of rLR over LR is statistically significant, as tested at the 5% level using a Wilcoxon Rank Sum test. This suggests that there is high potential for learning from unreliable data from the Internet using the label-noise robust algorithm proposed.

6 Conclusions

We proposed an efficient algorithm for learning a label-robust logistic regression algorithm for both two-class (rLR) and multiclass (rmLR) classification problems, and we proved its local convergence. We also developed a Bayesian sparse regularised extension for these methods which bypasses the need to perform cross validation for model selection and is hence label-robust in its model selection procedure as well. We demonstrated the working and the advantages of our approach in both controlled synthetic settings and in real applications. In particular, we have seen an application in the biomedical domain, where our method can be used to flag suspicious labels for further follow-up study. We have also seen that the label-flipping probabilities provide an interpretable holistic graphical view of data sets by unearthing the topology that underlies the data classes. Finally, the model can be used to facilitate the task of annotating training examples since it is now possible to learn the classifier from sloppily labelled but cheaply obtained data from crowds. Extending this approach to non-linear classifiers is the subject of our future work.

References

1. Alon, U., Barkai, N., Notterman, D.A., Gishdagger, K., Ybarradagger, S., Mackdagger, D., Levine, A.J.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proceedings of the National Academy of Sciences of the United States of America 96(12), 6745–6750 (1999)
2. Barandela, R., Gasca, E.: Decontamination of Training Samples for Supervised Pattern Recognition Methods. In: Amin, A., Pudil, P., Ferri, F., Iñesta, J.M. (eds.) SPR 2000 and SSPR 2000. LNCS, vol. 1876, pp. 621–630. Springer, Heidelberg (2000)
3. Brodley, C.E., Friedl, M.A.: Identifying mislabeled training data. Journal of Artificial Intelligence Research 11, 131–167 (1999)
4. Cawley, G.C., Talbot, N.L.C.: Gene selection in cancer classification using sparse logistic regression with bayesian regularization. Bioinformatics/Computer Applications in The Biosciences 22, 2348–2355 (2006)
5. Cawley, G.C., Talbot, N.L.C.: Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters. J. Mach. Learn. Res. 8, 841–861 (2007)
6. Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., Haussler, D.: Support vector machine classification and validation of cancer tissue samples using microarray expression data. Bioinformatics, 906–914 (2000)
7. Hausman, J.A., Abrevaya, J., Scott-Morton, F.M.: Misclassification of the dependent variable in a discrete-response setting. Journal of Econometrics 87(2), 239–269 (1998)
8. Hestenes, M.R., Stiefel, E.: Methods of Conjugate Gradients for Solving Linear Systems. Journal of Research of the National Bureau of Standards 49(6), 409–436 (1952)
9. Jiang, Y., Zhou, Z.-H.: Editing Training Data for *k*NN Classifiers with Neural Network Ensemble. In: Yin, F.-L., Wang, J., Guo, C. (eds.) ISNN 2004. LNCS, vol. 3173, pp. 356–361. Springer, Heidelberg (2004)

10. Kabán, A., Tiňo, P., Girolami, M.: A General Framework for a Principled Hierarchical Visualization of Multivariate Data. In: Yin, H., Allinson, N.M., Freeman, R., Keane, J.A., Hubbard, S. (eds.) IDEAL 2002. LNCS, vol. 2412, pp. 518–523. Springer, Heidelberg (2002)
11. Kadota, K., Tominaga, D., Akiyama, Y., Takahashi, K.: Detecting outlying samples in microarray data: A critical assessment of the effect of outliers on sample classification. Chem. Bio. Informatics Journal 3(1), 30–45 (2003)
12. Krishnan, T., Nandy, S.C.: Efficiency of discriminant analysis when initial samples are classified stochastically. Pattern Recognition 23(5), 529–537 (1990)
13. Lawrence, N.D., Schölkopf, B.: Estimating a kernel fisher discriminant in the presence of label noise. In: Proceedings of the 18th International Conference on Machine Learning, pp. 306–313. Morgan Kaufmann (2001)
14. Lee, D.D., Seung, H.S.: Algorithms for Non-negative Matrix Factorization. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems, vol. 13, pp. 556–562. MIT Press (2001)
15. Li, L., Darden, T.A., Weingberg, C.R., Levine, A.J., Pedersen, L.G.: Gene assessment and sample classification for gene expression data using a genetic algorithm / k-nearest neighbor method. In: Combinatorial Chemistry and High Throughput Screening, pp. 727–739 (2001)
16. Liu, Z., Jiang, F., Tian, G., Wang, S., Sato, F., Meltzer, S.J., Tan, M.: Sparse logistic regression with lp penalty for biomarker identification. Statistical Applications in Genetics and Molecular Biology 6(1), 6 (2007)
17. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision, ICCV 1999, vol. 2, pp. 1150–1157. IEEE Computer Society, Washington, DC (1999)
18. Lugosi, G.: Learning with an unreliable teacher. Pattern Recogn. 25, 79–87 (1992)
19. Mackay, D.J.C.: Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks. Network: Computation in Neural Systems 6, 469–505 (1995)
20. Magder, L.S., Hughes, J.P.: Logistic regression when the outcome is measured with uncertainty. American Journal of Epidemiology 146(2), 195–203 (1997)
21. Malossini, A., Blanzieri, E., Ng, R.T.: Detecting potential labeling errors in microarrays by data perturbation. Bioinformatics 22(17), 2114–2121 (2006)
22. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7), 971–987 (2002)
23. Raykar, V.C., Yu, S., Zhao, L.H., Valadez, G.H., Florin, C., Bogoni, L., Moy, L.: Learning from crowds. Journal of Machine Learning Research 11, 1297–1322 (2010)
24. Roth, V.: The generalized lasso. IEEE Transactions on Neural Networks 15, 16–28 (2004)
25. Yasui, Y., Pepe, M., Hsu, L., Adam, B.L., Feng, Z.: Partially supervised learning using an emboosting algorithm. Biometrics 60(1), 199–206 (2004)
26. Zhang, C., Wu, C., Blanzieri, E., Zhou, Y., Wang, Y., Du, W., Liang, Y.: Methods for labeling error detection in microarrays based on the effect of data perturbation on the regression model. Bioinformatics 25, 2708–2714 (2009)

Sentiment Classification with Supervised Sequence Embedding

Dmitriy Bespalov[1], Yanjun Qi[2], Bing Bai[2], and Ali Shokoufandeh[1]

[1] Drexel University,
Philadelphia, PA USA
[2] NEC Labs America,
Princeton, NJ USA

Abstract. In this paper, we introduce a novel approach for modeling n-grams in a latent space learned from supervised signals. The proposed procedure uses only unigram features to model short phrases (n-grams) in the latent space. The phrases are then combined to form document-level latent representation for a given text, where position of an n-gram in the document is used to compute corresponding combining weight. The resulting two-stage supervised embedding is then coupled with a classifier to form an end-to-end system that we apply to the large-scale sentiment classification task. The proposed model does not require feature selection to retain effective features during pre-processing, and its parameter space grows linearly with size of n-gram. We present comparative evaluations of this method using two large-scale datasets for sentiment classification in online reviews (Amazon and TripAdvisor). The proposed method outperforms standard baselines that rely on bag-of-words representation populated with n-gram features.

Keywords: Sentiment Classification, Large-Scale Text Mining, Supervised Feature Learning, Supervised Embedding.

1 Introduction

In this paper, we consider the problem of sentiment classification (SC) which is defined as identifying and extracting subjective information from natural language text. Due to the widespread use of electronic media and the explosion of online social-oriented content such as user reviews, sentiment classification [1], has received significant attention in recent years. This task aims to rate polarity of a given text accurately towards a label, predicting whether the expressed opinion in the text is positive, negative, or neutral.

SC task can be viewed as an instance of text categorization (TC) task. Notable varieties of TC include single-label, multi-label [2] or taxonomic hierarchy of labels [3]. Both generative approaches [4–6] and discriminative supervised methods have been applied to TC, and a few semi-supervised attempts [7] as well. Among discriminative models, support vector machines (SVM) are known for their superior performance in TC, and SC task [8, 9] in particular. Previous works on the discriminative TC commonly rely on the so-called *bag-of-words* (BoW) representation that maps variable length text into a fixed-dimensional vector, parameterized by a finite vocabulary. The

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 159–174, 2012.

"bag-of-unigrams" is the most common form of BoW representation utilizing a dictionary of basic words as its vocabulary. Essentially BoW model treats a document as an unordered collection of word-features, and utilizes the frequency distribution of the words as the primary evidence for TC.

There has been increasing evidence that short phrases are more effective than single words (unigrams) for the SC task. For example, the term *good* often appears in positive online reviews, but "not very good" is less likely to appear in positive comments. When using bag-of-unigrams representation, the proximity of "not", "good" and "very" in the text is ignored. A proposed remedy is to extend the bag-of-unigrams model by incorporating n-grams (a contiguous sequence of n words) [1] as features in the vector space representation of the text [10], i.e. so called "bag-of-n-grams" (BoN). However, extending the model to incorporate n-grams (for $n >= 3$) will adversely effect the complexity of parameter space, since the dimensionality of a BoN vector grows exponentially as a function of n. For instance, extending an English word vocabulary $\mathcal{D}$ of size $|\mathcal{D}| = 10,000$ by including the bigrams ($n = 2$) and trigrams ($n = 3$) will add up to $|\mathcal{D}|^2 = 10^8$ and $|\mathcal{D}|^3 = 10^{12}$ additional free parameters, respectively. Feature selection (FS) techniques [11] are popular methods for dealing with the complexity of bag-of-n-grams model. The basic idea of FS is to retain a small subset of features, based on a certain scoring function (statistics), that are suitable for the SC task. Popular FS methods used in classifying text include Information Gain (IG), Chi-Square Test (CHI), Mutual Information (MI), Optimal Orthogonal Centroid feature selection (OCFS) [11]. More recently, Jing *et al.* [12] introduced a generalized framework for popular FS techniques. However, effectiveness of FS methods is often dataset-dependent. Thus, choosing an appropriate FS technique requires empirical validation. Furthermore, estimating optimal hyper-parameters for each FS method considered will require additional cross-validation.

Our work is motivated by the idea of utilizing short phrases as features for large-scale sentiment classification. However, in contrast to BoN model, we propose a different approach to modeling short phrases for SC task. The proposed method projects n-grams into a latent lower-dimensional space using only unigram features and avoids FS preprocessing. To be more specific, the embedding of an n-gram is a combination of the embedding of its composing words. The procedure estimates an embedding of a unigram feature for every position in the n-gram window. In this way, the parameter space of our model grows only linearly with n. The embedding of the whole document is a union of such n-gram embeddings, re-weighted based on their positions in the text. This is consistent with the hypothesis that spatial occurrence of a phrase can influence overall sentiment of an article. The parameters of our method are jointly optimized in an online learning setting through the stochastic gradient descent method [13]. The empirical evaluations demonstrate the proposed model outperforming state-of-art FS method for the SC task.

To summarize, the proposed system performs feature selection in a latent space, promoting phrases that are effective for the SC task. Section 3.3 provides interesting anecdotal evidence to support this claim. We evaluate the performance of the proposed method along with standard baselines on two sentiment classification datasets

[1] We will use "n-gram" and "phrase" interchangeably.

(Amazon[2], TripAdvisor[3]). Our empirical evidence demonstrates that the proposed framework outperforms baseline methods.

2 Supervised Sequence Embedding

In this section we will present an overview of the proposed deep neural network model that 1) represents all sliding n-gram windows in a lower-dimensional latent space; 2) obtains document representation in the latent space, defined as a weighted sum of latent n-grams, where weights are learned from positions of phrases in the document; and 3) estimates a classifier in the document-level latent space, biased towards the prescribed classification task.

Before presenting the proposed "deep" neural network model, an overview of the notations is in order. Let $\mathcal{D}$ denote the underlying word (unigram) dictionary and $\mathcal{S}$ denote the set of all finite length sequences of words from $\mathcal{D}$. We use $\Gamma = \Gamma(n) \subset \mathcal{S}$ to denote the vocabulary of n-grams in a text corpus. An input text $\mathbf{x} \in \mathcal{S}$ of length N is an ordered sequence $\mathbf{x} = (w_1, \ldots, w_N)$, with $w_i \in \mathcal{D}$. We denote an n-gram from $\mathbf{x}$, with $n < N$, starting at its j-th position as $\gamma_j = (w_j, w_{j+1}, \ldots, w_{j+n-1})$. We denote vectors or matrices with boldface font (e.g., $\mathbf{G}$ or $\mathbf{b}$), and use cursive for scalar variables and functions (e.g., M or $h(\cdot)$). We use $|.|$ to denote the cardinality of a set. Operator $\times$ denotes vector or matrix multiplication, while $\cdot$ will be used to emphasize the multiplication of scalar variables. Let $\mathcal{Y} = \{1, ..., C\}$ denote a set of class labels. $\mathcal{X} \subset \mathcal{S}$ denotes a collection of labeled documents (training set), where $\mathcal{X} = \{(\mathbf{x}_i, y_i)_{i=1,\ldots,L} | \mathbf{x}_i \in \mathcal{X} \;\&\; y_i \in \mathcal{Y}\}$ and $|\mathcal{X}| = L$.

Our model is an alternative to the classification with BoW representation. For text $\mathbf{x} = (w_1 \cdots w_N)$, the BoW model uses a unigrams dictionary to produce a $|\mathcal{D}|$-dimensional vector $\tilde{\mathbf{e}}_{\mathbf{x}}$ for $\mathbf{x}$:

$$\tilde{\mathbf{e}}_{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{e}_{w_i}, \tag{1}$$

Here $\mathbf{e}_{w_i}$, also known as "selector", is the canonical basis vector

$$\mathbf{e}_{w_i} = (0, \ldots, 0, \underset{\text{at index } w_i}{1}, \ldots, 0)^\top \tag{2}$$

with a single non-zero entry at w_i-th position. It is a common practice to replace the sole non-zero entry of $\mathbf{e}_{w_i}$ with the inverse document frequency of word w_i. As a result, the vector $\tilde{\mathbf{e}}_{\mathbf{x}}$ in (1) takes the form of TF-IDF weighting.

The BoN extension includes n-grams as additional features [14]. By using all unique phrases of at most n words from Γ as features, we obtain a $|\Gamma|$-dimensional representation of $\mathbf{x}$. That is, BoN maps $\mathbf{x}$ to $|\Gamma|$-dimensional representation, with $|\Gamma| = O(|\mathcal{D}|^n)$. Due to its exploding number of features, BoN normally relies on feature selection methods to control the number of parameters. Differently, our method models n-grams in the latent space while recognizing only $n \cdot |\mathcal{D}|$ unique features and avoiding feature selection pre-processing.

[2] http://times.cs.uiuc.edu/~wang296/Data/TripAdvisor.tar.gz

[3] http://www.cs.jhu.edu/~mdredze/datasets/sentiment/unprocessed.tar.gz

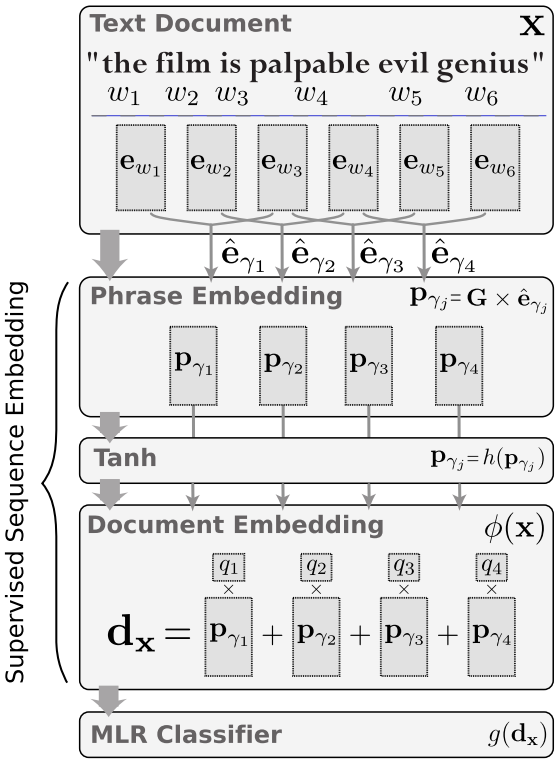


Fig. 1. Sentiment Classification using Supervised Sequence Embedding. We consider two variations of the method based on the procedure to compute combining weights q_j.

We refer to the first two stages of the proposed system as the "Supervised Sequence Embedding" (SSE). Figure 1 provides an illustration of the SC system. The first projection step that computes latent embedding of all n-grams in an article is presented in Section 2.1. The second projection that combines latent n-grams to compute article-level embedding is presented in Section 2.2. Section 2.3 presents the third step of the SC system that computes a document-level classifier in the latent space. Our framework is best described using a multi-layer projection as shown in Figure 1.

2.1 Latent n-Gram Embedding

We formally define the projection step for latent phrase embedding. The formation of n-grams is carried through a sliding window of length n. As illustrated in Figure 1, setting $n = 3$, the first n-gram is (w_1, w_2, w_3), second n-gram (w_2, w_3, w_4), etc. Given a phrase of n adjacent words, we first represent it using n word selectors. Specifically, given $\gamma_j = (w_j, w_{j+1}, \ldots, w_{j+n-1})$, define

$$\hat{\mathbf{e}}_{\gamma_j} = [\mathbf{e}_{w_j}^\top, \mathbf{e}_{w_{j+1}}^\top, \ldots, \mathbf{e}_{w_{j+n-1}}^\top]^\top, \tag{3}$$

where the notation $[\cdot]$ denotes the concatenation of single word selectors into an $n \cdot |\mathcal{D}|$-dimensional vector for each n-gram γ_j. The embedding of the γ_j is then defined as:

$$\mathbf{p}_{\gamma_j} = \mathbf{G} \times \hat{\mathbf{e}}_{\gamma_j}, \tag{4}$$

where $\mathbf{G} \in \mathbb{R}^{M \times n \cdot |\mathcal{D}|}$ is the projection matrix which maps $\hat{\mathbf{e}}_{\gamma_j}$ into a latent space with dimension M. It is important to note that M is a hyperparameter, while parameters of $\mathbf{G}$ are estimated during the learning process using backpropagation.

Since each n-gram is encoded as a sparse vector with n non-zeros in $\hat{\mathbf{e}}_{\gamma_j}$, we can treat (4) as an operation decoupling the embedding parameters for word w_i based on its position with the n-gram γ_j. Matrix $\mathbf{G}$ maintains n latent embedding vectors for every word $w_i \in \mathcal{D}$ depending on its position inside the n-gram. That means one embedding for each possible position within the n-gram for w_i.

2.2 Latent Document Embedding

We use the n-gram embedding to form a vector representation for a text document. The number of phrases in each document is variable depending on its length N. We need a function to compress the information from these n-grams into a fixed length document embedding vector. While there are many possibilities for the combining function, the mean$(\cdot)$ function has been verified by our previous work in [15] to provide a good summarization of a document in the latent space. In this work, we also propose to use a weighted sum function and to learn the weights for each $\gamma_j \in \mathbf{x}$, based on its position in the text. These weights are used to combine latent embedding of γ_j into a document-level representation. Specifically, we define latent embedding of document $\mathbf{x}$ in the latent space as:

$$\phi(\mathbf{x}) \equiv \mathbf{d}_{\mathbf{x}} = \sum_{j=1}^{N} q_j \times h(\mathbf{p}_{\gamma_j}), \tag{5}$$

where $\mathbf{d}_{\mathbf{x}} \in \mathbb{R}^M$, $\mathbf{x} = (w_1, \ldots, w_N)$, and $h(\cdot) = \tanh(\cdot)$[4]. We model the weight of every γ_j using the following mixture model. Let $\gamma_j \in \mathbf{x}$, $|\mathbf{x}| = N$ and j indicate the position of an n-gram in $\mathbf{x}$, and define the weight associated with γ_j as:

$$q_j = \frac{1}{Q} \sum_{k=1}^{K} \text{sigmoid}\left(a_k \cdot \frac{j}{N} + b_k\right), \tag{6}$$

where a_k, b_k are parameters to be learned, $Q = \sum_{j=1}^{N} q_j$, K specifies the number of mixture quantities, and sigmoid$(\cdot)$ is a non-linear transfer function. In the rest of this manuscript, we refer to the model with uniform weights $q_j = \frac{1}{N}$ (i.e., combining function is mean$(\cdot)$) as SSE, while SSE-W is used to denote the model with spatial re-weighting defined in (6). In spatial re-weighting in SSE-W model, it attempts to capture longer "trends" within each document. In this, our work is similar to the work of Lebanon *et al.* [16]. The authors propose a novel semi-parametric generative model for an unsupervised embedding of documents as smooth curves in $\mathbb{R}^{|\mathcal{D}|}$, while preserving spatial information for phrases within a document.

[4] The non-linear function $\tanh(\cdot)$ converts the unbounded range of the input into $[-1, 1]$.

2.3 Classifier

In our evaluations we use Multinomial Logistic Regression (MLR) to carry out SC. Given the document embedding $\mathbf{d_x}$, and C candidate classes, $\boldsymbol{\beta}_i$ represents the coefficient weights for the i-th candidate class. Furthermore, the predicted class label can be calculated as follows:

$$g(\mathbf{x}) = \underset{i \in \{1..C\}}{\arg\max} \frac{\exp(\boldsymbol{\beta}_i^\top \times \mathbf{d_x})}{1 + \sum_{k \in \{1..C\}} \exp(\boldsymbol{\beta}_k^\top \times \mathbf{d_x})} \tag{7}$$

This classifier can be trained by minimizing the loss function:

$$\mathcal{L}(\mathcal{X}) = - \sum_{i \in \{1..|\mathcal{X}|\}} \log \frac{\exp(\boldsymbol{\beta}_{y_i}^\top \times \mathbf{d}_{\mathbf{x}_i})}{1 + \sum_{j \in \{1..C\}} \exp(\boldsymbol{\beta}_j^\top \times \mathbf{d}_{\mathbf{x}_i})} \tag{8}$$

This latter loss is called "negative log likelihood" in literature.

The proposed supervised embedding method is implemented as a perceptron network composed of four activation layers as shown in Figure 1. We take advantage of the backpropagation process to train this layered network and use stochastic gradient descent (SGD) method for estimating the parameters [13]. For a training set $\mathcal{X}$, instead of calculating true gradient of the objective with all training samples, SGD computes the gradient with a randomly chosen training sample and updates all parameters accordingly. SGD optimization method is scalable and proven to rival the performance of batch-mode gradient descent methods when dealing with large-scale datasets [17].

2.4 Related Methods

The proposed SSE embedding has its roots in a previous model known as "Lookup Table Convolution" (LTC) [15, 18]. LTC constructs a low-dimensional latent embedding for all $\gamma_j \in \mathbf{x}$ by first projecting each word into a latent space, followed by a second projection step to obtain the latent embedding of each n-gram. Specifically, each word $w_j \in \mathcal{D}$ is embedded into the m-dimensional feature space using a word lookup table:

$$LT_{\mathbf{E}}(w_j) = \mathbf{E} \times \mathbf{e}_{w_j} = \mathbf{E}_{w_j}, \tag{9}$$

where the j-th column of the matrix $\mathbf{E} \in \mathbb{R}^{m \times |\mathcal{D}|}$ denotes the embedding vector of the word w_j. Given an n-gram γ_j, the word lookup table applies the same operation to each word inside the n-gram sliding window, producing the vector $\mathbf{z}_{\gamma_j} = [\mathbf{E}_{w_j}^\top, \mathbf{E}_{w_{j+1}}^\top, \ldots, \mathbf{E}_{w_{j+n-1}}^\top]^\top$, with $[\cdot]$ denoting the concatenation of single word embedding into an $n \cdot m$-dimensional vector. The latent embedding for γ_j is then defined as

$$\tilde{\mathbf{p}}_{\gamma_j} = \mathbf{F} \times \mathbf{z}_{\gamma_j} = \mathbf{F} \times [\mathbf{E}_{w_j}^\top, \mathbf{E}_{w_{j+1}}^\top, \ldots, \mathbf{E}_{w_{j+n-1}}^\top]^\top, \tag{10}$$

where projection matrix $\mathbf{F} \in \mathbb{R}^{M \times n \cdot m}$ maps $\mathbf{z}_{\gamma_j}$ into the M-dimensional latent space. This two-step embedding procedure encodes each n-gram in a latent space without the explicit construction of all n-grams. Collobert and Weston [18] empirically validated LTC on six Natural Language Processing (NLP) tasks. Our previous work [15] adopted LTC for sentiment classification.

We emphasize the difference between this work and [15]. First, instead of modeling a lookup table layer followed by a convolutional layer as done in LTC, SSE models the parameters of the latent n-gram embedding directly using matrix G in (4). Second, the SSE-W model uses spatial re-weighting of n-grams, while uniform weights (i.e., mean$(\cdot)$ combining function) are used in LTC (and SSE). This improves the performance in many cases as our experimental evaluations in Section 3 suggest. In addition, the experimental results suggest the SSE model achieves higher SC accuracy, compared to the LTC method described by (9) and (10). Furthermore, training an LTC model using backpropagation requires many vector multiplications to calculate gradients $\partial\mathcal{L}/\partial\mathbf{E}$ and $\partial\mathcal{L}/\partial\mathbf{F}$ due to the multiplicative coupling of $\mathbf{E}$ and $\mathbf{F}$. In contrast, in training SSE models, these computations are largely avoided.

In general, performing dimensionality reduction in the original high-dimensional feature space is a common practice for various classification methods. Popular unsupervised latent embedding methods on text documents includes Latent Semantic Indexing (LSI) [19], or its probabilistic extensions, probabilistic LSI (pLSI) [20], and Latent Dirichlet Allocation (LDA) [21]. However, biasing parameters of the embedding towards specific classification task has not received much attention until recently, such as LTC from [18] and the work of learning to rank with joint word-image embedding in [22].

Our work is also related to the "Deep learning" architecture which has received increasing attention in recent years. Deep architectures have been used to learn complicated functions in natural language processing and computational vision [23]. Each layer in the architecture encodes features at different levels of abstraction, defined as a composition of features computed at the previous layer. Glorot *et al.* [24] utilize a deep learning model to extract the representation of each text review in an unsupervised fashion using stacked Denoising Auto-encoders. With the learned high-level feature representation the authors claim to achieve state-of-the-art performance for domain adaption tasks on sentiment classification data. Socher *et al.* [25] use recursive neural networks to perform simultaneous parsing and classification of both text and image data. In addition, multi-layered neural networks are successfully used for learning language models that estimate conditional probability distribution for word sequences [26, 27].

Another relevant domain to our work is the "string kernel" framework. String kernels and their extensions have been very popular discriminative choices for the protein classification problem, where sequences of amino acids are represented as strings [28, 29]. These kernels map a variable length string into a low-dimensional dense feature space using BoW strategy. Similar approaches have also been applied to text categorization before: see e.g., [30]. A critical component in the string kernel research is the implementation of inexact matching between short sequence segments. These approaches give rise to a family of mismatch kernels [28]. Similarly, the SSE method allows for inexact phrase matching that takes place in the latent space.

3 Experiments

We evaluate the performance of the proposed SSE method on SC task with binary and multi-class setting. For binary classification setting, we only consider positive (1 and 2 stars) or negative (4 and 5 stars) sentiment in the reviews. For multi-class setting we use

four available labels (1,2,4 and 5 stars) to evaluate text classification on Amazon and TripAdvisor datasets. In addition, since TripAdvisor contains neutral reviews, we also consider a SC task with five category labels for this dataset.

Amazon dataset contains customer reviews of 25 various categories of goods including apparel, automotive, baby, DVDs, electronics, magazines, and tools and hardware. TripAdvisor dataset contains customer reviews for various hotels across the globe. While TripAdvisor corpus provides rating scores for various aspects (e.g., rooms, location, cleanliness), we only consider overall ratings for this dataset. These are considered some of the largest sentiment classification datasets currently available. For Amazon we use 257,900 samples for training and 110,562 samples for testing, while 55,306 and 10,078 samples from TripAdvisor were used for training and testing, respectively. The development sets contain 10,000 and 5,000 samples for Amazon and TripAdvisor, respectively. In this work, we report classification results obtained using train-development-test splits for Amazon and TripAdvisor datasets. These dataset splits are available for download from our website[5]. It is important to note that the empirical evidence reported in this work are not directly comparable to the results we reported in [15], as we use different splits for Amazon and TripAdvisor datasets. However, to be fair, we make available online the SA results for the proposed SSE method, benchmarked on the split used in our previous publication [15].

Amazon and TripAdvisor datasets contain user-generated reviews where an overall sentiment for each review is quantified with an integer 1 through 5 (a.k.a the 5-star Likert scale). A sentiment score of 1 star corresponds to the lowest (negative) sentiment, while the score of 5 stars corresponds to the highest (positive) sentiment. TripAdvisor dataset contains neutral reviews (rated with 3-stars), while neutral reviews were omitted during the construction of Amazon dataset by their authors. For both datasets, a balanced version of the data splits (i.e., training / testing / development) is created that contain equal number of positive (4 and 5 stars) and negative (1 and 2 stars) reviews.

Table 1 provides the number of unique phrases for $n \in \{1, 2, 3, 5\}$ found in the training sets. Clearly, when $n \geq 2$, a feature selection technique is necessary, not only to improve the classification accuracy but also to keep the optimization tractable. For both datasets, we follow the method used in [31] to limit the vocabulary size by retaining n-grams with the highest mutual information (MI) shared by the binary labels (positive or negative). For the Amazon dataset, we use training split to select 25,000 grams per category, then concatenate the phrases to form the vocabulary used in the experiments. For TripAdvisor dataset, we also use an MI-based procedure to limit the vocabulary size to 500,000 n-grams computed for the entire training corpus.

For one of the baseline methods we use a linear SVM classifier, which is trained on BoN document representation. We obtain BoN representation with TF-IDF and $n \in \{1, 2, 3, 5\}$. We restrict our evaluation to the linear kernel because of the corpus size and the number of features used in describing each document. Prior research showed linear SVM achieving state-of-art performance on SC tasks (see e.g., [8] or [9]). In addition, we use a linear perceptron classifier trained on BoN as another baseline. We believe the latter choice is relevant, since the main objective of this work is to test the merit of the proposed SSE against the BoW model populated with n-grams.

[5] http://mst.cs.drexel.edu/datasets/ECML2012

Table 1. Unique phrase counts $|\mathbf{\Gamma}|$ for each dataset. Numbers are in thousands.

n-gram size	Amazon	TripAdvisor	RCV1 23k	RCV1 380k
$n = 1$	448	158	124	262
$n = 2$	6,446	1,175	2,400	6,364
$n = 3$	23,400	5,172	9,535	30,377
$n = 5$	78,864	21,741	35,118	262,586

We use **SVM** and **Prc** to denote the SVM and linear perceptron classifiers, respectively. The BoN representation will be denoted with **BoW-ng**, while $|\mathbf{\Gamma}|$ denotes the number of unique phrases (in thousands) for the training sets. We use **LTC** for referring to the Lookup Temporal Convolution method presented in [15]. We denote the proposed SSE method with $mean(\cdot)$ used for combining function as **SSE**. SSE method with spatial re-weighting of n-grams defined in (6) is identified as **SSE-W**. The rest of this section is organized as follows. We provide important implementation details in Section 3.1. Sentiment classification results are discussed in Section 3.2. Section 3.3 provides anecdotal evidence that selecting phrases with highest prediction responses from a trained SSE model can be used for "sentiment summarization". We also provide an illustration of the estimated spatial weights for trained SSE-W model. In addition we demonstrate that SSE-W can be augmented with an alternative combining function q_j that captures strength of sentiment in text, but not polarity. Finally, in Section 3.4 we present topic categorization results on Reuters dataset [32] to demonstrate that SSE model is applicable to text categorization tasks other than SC.

3.1 Implementation Details

In our implementation we used the following formulation of TF-IDF. For every n-gram $\gamma_j \in \mathbf{x}$ where document $\mathbf{x} \in \mathcal{X}$, the weight for γ_j was calculated using the formula: $\text{tfidf}(\gamma_j, \mathbf{x}, \mathcal{X}) = \frac{1}{|\mathbf{x}|} \cdot \text{tf}(\gamma_j, \mathbf{x}) \cdot \text{idf}(\gamma_j, \mathcal{X})$, where $\text{idf}(\gamma_j, \mathcal{X}) = \log \frac{|\mathcal{X}|}{|\{\mathbf{x}_i \in \mathcal{X} : \gamma_j \in \mathbf{x}_i\}|}$, and $\text{tf}(\gamma_j, \mathbf{x})$ returns the number of times term γ_j appears in $\mathbf{x}$.

We used the LIBLINEAR[6] SVM toolkit. For each SC task, the penalty parameter C was set using grid search with $C = \{2^{-8}, 2^{-7}, \ldots, 2^{10}, 2^{11}\}$, performed on the development set. Then the reported classification error was computed on the testing set with the optimal penalty parameter found. Perceptron-based methods (LTC, SSE, SSE-W and Prc BoW) were implemented using the Torch5[7] machine learning library. A development set was used to select the best model during the training of all perceptron classifiers. During the training procedure the model was evaluated at regular intervals on the entire development set, and the best performing model was retained. After the training was completed, this model was used to compute the classification error rate for the testing set, which is the number reported in all of our experiments below.

The perceptron classifiers were trained with a fixed learning rate 0.05. The dimensionality of the latent space for all perceptron-based methods was set to $M = 50$ in all

[6] http://www.csie.ntu.edu.tw/~cjlin/liblinear/
[7] http://torch5.sourceforge.net/

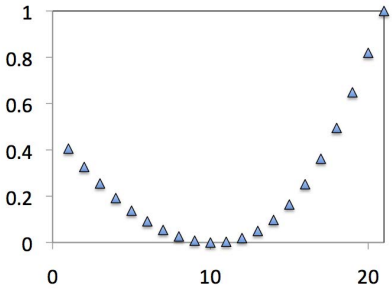
Table 2. Macro-average classification error rate for SVM with BoN, where $n \in \{1, 2, 3, 5\}$. Macro-average error rate is calculated as mean of per-label classification error rates. $2 \cdot \star$ denote binary classification setting, while $4 \cdot \star$ and $5 \cdot \star$ identify multi-class setting with four and five categories, respectively. The numbers marked with † (or ‡) are statistically significantly better than **SVM BoW-1g** with $p < 0.0001$ (or $p < 0.01$).

Method	Amazon		TripAdvisor		
	$2 \cdot \star$	$4 \cdot \star$	$2 \cdot \star$	$4 \cdot \star$	$5 \cdot \star$
SVM BoW-1g	10.68	35.78	8.97	35.41	46.41
SVM BoW-2g	6.60†	28.26†	7.60‡	33.68‡	**44.68**‡
SVM BoW-3g	**6.39**†	**27.98**†	**7.46**‡	33.50‡	45.12
SVM BoW-5g	6.48†	28.02†	7.53‡	**33.45**‡	46.41

the experiments. We set these parameters according to our prior experience in designing perceptron-based classification systems, and did not subject them to the empirical selection in this work. We selected the length of the latent phrase for SSE and SSE-W methods (i.e., size of n-gram) after evaluating SVM classification performance with BoN and $n \in \{1, 2, 3, 5\}$. These results are presented in Table 2. We selected $n = 5$ when modeling latent phrases in SSE and SSE-W methods. Finally, we fixed $K = 3$ in (6), which was motivated by our assumption that phrases appearing in the beginning or at the end of each text are the most effective at predicting text labels. The results presented in Section 3.3 support this hypothesis.

3.2 Classification Results

Table 3 presents SC results using macro-average error rate, defined as mean of per-label classification error rates. For completeness of presentation, we also provide micro-average classification error rates in Table 4, computed over all test samples regardless



(a) SSE-W with spatial weights (6)

5-gram	Weight
is an extremely good book	3.58
is just a good book	3.19
book is a good buy	2.94
overall a very good book	2.84
book is a good choice	2.81
book is a very good	2.76
book is still very good	1.70
a good book just because	1.15
unless good books are just	1.05

(b) SSE-W with weights $q_j = \hat{\mathbf{G}} \times \hat{\mathbf{e}}_{\gamma_j}$

Fig. 2. (a) Illustration of spatial weights in SSE-W model trained on the Amazon dataset. The values of the spatial weights were computed for a "synthetic" text with 25 words. The weights are scaled into range $[0, 1]$ for illustration purposes. **(b)** Select 5-grams and their combining weights. The weights are computed using the model $q_j = \hat{\mathbf{G}} \times \hat{\mathbf{e}}_{\gamma_j}$ trained on Amazon dataset with binary classification setting.

of their labels. It is worth noting that splits for Amazon and TripAdvisor are balanced in terms of binary sentiment polarity, thus binary classification error rates in Table 4 match the macro-average results from Table 3. In the five experiments conducted SSE-W method outperforms the SVM baseline. However, only in the multi-class setting SSE-W method results in statistically significant improvement over SSE model, with $p < 0.0001$ for Amazon and $p < 0.01$ for TripAdvisor datasets. In case of binary classification on Amazon, improvement of SSE-W over SSE is only statistically significant with $p < 0.4$. These results suggest that spatial re-weighting of phrases only becomes relevant when predicting sentiment on the Likert scale with multiple labels. Also, when predicting binary sentiment, the presence of certain phrases, regardless of their positions within the text, is sufficient for the task.

Table 3. Macro-average classification error rate. Macro-average error rate is calculated as mean of per-label classification error rates. $2 \cdot \star$ denote binary classification setting, while $4 \cdot \star$ and $5 \cdot \star$ identify multi-class setting with four and five categories, respectively. The numbers marked with † (or ‡) are statistically significantly better than **SVM BoW-3g** with $p < 0.0001$ (or $p < 0.01$).

Method	Amazon		TripAdvisor		
	$2 \cdot \star$	$4 \cdot \star$	$2 \cdot \star$	$4 \cdot \star$	$5 \cdot \star$
SVM BoW-3g	6.39	27.98	7.46	33.50	45.12
Prc BoW-3g	6.55	26.45†	7.54	34.73	43.58
SSE	5.69†	25.30†	**6.90**	34.22	42.88‡
SSE-W	**5.63**†	**24.61**†	7.01	**32.25**	**40.54**†
LTC	7.05	-	8.49	-	-

3.3 Illustrative Examples

We now present several illustrative examples obtained using SSE and SSE-W models trained on the sentiment datasets with multi-class $4 \cdot \star$ setting. Table 5 shows three non-overlapping 5-grams with highest weights obtained from selected TripAdvisor reviews. The weight for each phrase γ_j is set using $\max_{i \in \{1 \ldots C\}} \boldsymbol{\beta}_i^\top \times \phi(\gamma_j)$, where $\phi(\gamma_j)$ denotes latent embedding of γ_j. The trained SSE model for TripAdvisor is used to compute embedding of each phrase γ_j separately.

Table 4. Micro-average classification error rate. Micro-average error rate, computed over all test samples, regardless of their labels is reported. The numbers marked with † (or ‡) are statistically significantly better than **SVM BoW-3g** with $p < 0.0001$ (or $p < 0.01$).

Method	Amazon		TripAdvisor		
	$2 \cdot \star$	$4 \cdot \star$	$2 \cdot \star$	$4 \cdot \star$	$5 \cdot \star$
SVM BoW-3g	6.39	23.45	7.46	32.00	43.07
Prc BoW-3g	6.55	23.00‡	7.54	33.94	43.05
SSE	5.69†	22.40†	**6.90**	33.90	42.21
SSE-W	**5.63**†	**22.05**†	7.01	**31.41**	**40.76**‡

We also present sample illustration of spatial weights obtained from the SSE-W model trained on the Amazon dataset with binary classification setting. The values of the spatial weights were computed for a "synthetic" text with 25 words. The weights are scaled into range $[0, 1]$ for illustration purposes. Please refer to Figure 2a for the illustration. We note the obtained weights have a straightforward interpretation – phrases or sentences that appear in the beginning or at the end of each review are more likely to express strong sentiment that defines the polarity of the review.

In addition to spatial information, weights q_j in (6) can be computed with other models. For example, latent projection layer from (4) can be used to compute weights q_j directly from the sequence features. In this case, another projection $\hat{\mathbf{G}}$ is estimated, where $\hat{\mathbf{G}} \in \mathbb{R}^{1 \times n \cdot |\mathcal{D}|}$ and $q_j = \hat{\mathbf{G}} \times \hat{\mathbf{e}}_{\gamma_j}$. To illustrate our point, we use the SSE model trained on the binary Amazon dataset to initialize the modified SSE-W model. The modified SSE-W model is then trained on the Amazon dataset, while keeping the projection parameters $\mathbf{G}$ unchanged. We then use weights $q_j = \hat{\mathbf{G}} \times \hat{\mathbf{e}}_{\gamma_j}$ to sort all 5-grams from the Amazon's testing set. Figure 2b lists several top-scoring 5-grams that we have selected from the sorted list of the 5-grams that contained words "good" and "book". We note that this model captures only sentiment strength and the obtained list contains phrases that carry both positive and negative sentiment. The estimated weights q_j, presented in Figure 2b, support this argument. For example, one can argue that "is an extremely good book" carries stronger (positive) sentiment than "book is a good choice", which in turn has stronger sentiment than "a good book just because".

Table 5. Summarization for select TripAdvisor reviews, obtained as the top three non-overlapping 5-grams. The trained TripAdvisor SSE model with multi-class $4 \cdot \star$ setting is used to calculate phrase weights.

Review Text	Rating	5-gram	Weight
disappointing choice this **is one of the worst** large hotels i have ever visited . the suite i had was filthy , and the food from room service **was barely edible (the** caesar salad was dangerously inedible) . there is no wifi . two lamps do not work . feels like a decrepit ocean liner . despite the view and the location , i would **avoid this place at all** cost .	⋆	is one of the worst avoid this place at all was barely edible (the	34.1 31.3 28.6
noisy air conditioning on NUMBERnd floor ! ! we stayed one night in the sand villa , in a room on the NUMBERnd floor overlooking the pool . the room was comfortable . there was a loud rumbling noise , seemingly from something like a big central air conditioner , that continued all night . it was about as loud as a plane during flight - - certainly not , but not pleasant either . the **staff was pleasant and helpful** , but because of the noise i **would not stay there again** .	⋆⋆	staff was pleasant and helpful noisy air conditioning on would not stay there again	30.6 22.1 20.6
very nice experience the frenchmen **is a very nice place** to stay . the rooms were decorated nicely and the courtyard with the **jacuzzi and pool were beautiful** . above all , the staff was probably the friendliest i ' ve ever encountered . very outgoing and pleasant . **the only bad thing i** could say about it is that the rooms were just a little small , but for a single person or a close couple , it was fine .	⋆ ⋆ ⋆⋆	the only bad thing i jacuzzi and pool were beautiful is a very nice place	17.3 16.7 16.3
stylish and great staff i stayed at the hotel globus in may NUMBER as a single female traveller . the room was small but very stylish and spotless . **the staff were all fantastic** and very friendly . **good breakfast and excellent location** for the railway station and easy reach of all florence ' s attractions . i ' m going back to florence in december and will be staying there again .	⋆ ⋆ ⋆ ⋆ ⋆	the staff were all fantastic stylish and great staff i good breakfast and excellent location	26.8 22.1 20.6

3.4 Topic Categorization

In addition to SC task we consider topic categorization using Reuters dataset (RCV1)[8]. The original Reuters Corpus (RCV1) contains train-test split of 23,149 and 781,265 documents, respectively. The documents in the RCV1 corpus are categorized with 103 topics. The main focus of our research is the development of text classification methods that can efficiently handle large-scale data. Thus, we also create a new split for RCV1 with 380,000 training samples. Following the procedure of Lewis *et al.* [32] we select documents with IDs between 2,286 and 383,792 for training in the new split. We denote the split with the original (smaller) training set with **RCV1 23k**, while the split with larger training set is denoted by **RCV1 380k**. To obtain a development set for both splits of RCV1, we randomly sample 10,000 documents from the corresponding testing sets. We restrict our evaluations to the four topics with the largest number of positive examples in the entire Reuters Corpus: CCAT (ALL Corporate-Industrial) GCAT (All Government and Social), MCAT (ALL Securities and Commodities Trading and Markets) and C15 (Corporate and Industrial Performance). For RCV1 dataset we limit the vocabulary size to the 500,000 most frequent n-grams selected using training set only.

Table 6. Macro-average classification error rate for RCV1 dataset. Macro-average error rate is calculated as mean of per-label classification error rates. The numbers marked with † (or ‡) are statistically significantly better than **SVM BoW-2g** with $p < 0.0001$ (or $p < 0.01$).

Method	RCV1 23k				RCV1 380k			
	CCAT	GCAT	MCAT	C15	CCAT	GCAT	MCAT	C15
SVM BoW-2g	5.82	5.42	5.60	7.62	**4.07**	4.47	3.95	4.93
SSE	5.74	4.79†	**4.41**†	6.21†	4.29	**3.81**†	**3.42**†	5.76
SSE-W	**5.71**‡	**4.70**†	4.45†	**5.50**†	4.15	**3.81**†	3.47†	**4.28**†

Table 6 presents text categorization results for RCV1 dataset. SSE-W method outperforms the SVM baseline in all but one experiment. In addition, SSE-W does not improve classification over the SSE model for the MCAT topic, and the classification improvements are rather small for the CCAT and GCAT topics. On the other hand, the improvement of SSE-W over the SSE method is statistically significant with $p < 0.0001$ in the case of C15 topic. We speculate these results can be attributed to the nature of the topics considered. Indeed, MCAT, CCAT and GCAT are high-level topics in RCV1, with each assigned to news articles that describe broad range of concepts. Furthermore, C15 identifies articles only related to corporate and industrial performance, thus allowing SSE-W model to identify the spatial distribution of the effective phrases, that C15 articles exhibit.

4 Conclusions and Future Work

This work presents a supervised method (SSE) for the latent embedding of n-grams. The experimental results show improved text classification performance over the

[8] We use raw text features, instead of stemmed words as used in the original RCV1 publication.

baseline classifiers trained on BoN models. In addition, the proposed extension to the model (SSE-W) incorporates the relative position of the phrases when forming latent representation of a document. SSE-W model improves sentiment classification accuracy over SSE model that uses uniform weights ($q_j = \frac{1}{N}$).

We limit our empirical evaluation in this work to document-level text classification, focusing on sentiment analysis problem. We believe the SSE model can also be applied to sequence classification in general, where a sequence is an ordered list of events that can be described using a finite set of features. In the future work, we plan to investigate the merit of the proposed system for various sequence classification tasks. For example, the task of classifying protein sequences [33] or query log sequences to identify human users [34]. In addition, we plan to consider applying the framework to other modalities where BoW representation is used. For instance, object recognition in images is another promising direction of our future work.

References

1. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2(1-2), 1–135 (2008)
2. Zhu, S., Ji, X., Xu, W., Gong, Y.: Multi-labelled classification using maximum entropy method. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2005, pp. 274–281. ACM, New York (2005)
3. Sun, A., Lim, E.P.: Hierarchical text classification and evaluation. In: Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM 2001, pp. 521–528. IEEE Computer Society, Washington, DC (2001)
4. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. In: AAAI 1998 Workshop on Learning for Text Categorization, vol. 752, pp. 41–48 (1998)
5. Nigam, K.: Using maximum entropy for text classification. In: IJCAI 1999 Workshop on Machine Learning for Information Filtering, pp. 61–67 (1999)
6. Yi, K., Beheshti, J.: A hidden markov model-based text classification of medical documents. J. Inf. Sci. 35, 67–81 (2009)
7. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using em. Mach. Learn. 39, 103–134 (2000)
8. Mirowski, P., Ranzato, M., LeCun, Y.: Dynamic auto-encoders for semantic indexing. In: Proceedings of the NIPS 2010 Workshop on Deep Learning (2010)
9. Paltoglou, G., Thelwall, M.: A study of information retrieval weighting schemes for sentiment analysis. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010, pp. 1386–1395. Association for Computational Linguistics, USA (2010)
10. Cavnar, W., Trenkle, J.: N-gram-based text categorization. Ann. Arbor. MI 48113(2), 161–175 (1994)
11. Yan, J., Liu, N., Zhang, B., Yan, S., Chen, Z., Cheng, Q., Fan, W., Ma, W.Y.: Ocfs: optimal orthogonal centroid feature selection for text categorization. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2005, pp. 122–129. ACM, New York (2005)

12. Jing, H., Wang, B., Yang, Y., Xu, Y.: A General Framework of Feature Selection for Text Categorization. In: Perner, P. (ed.) MLDM 2009. LNCS, vol. 5632, pp. 647–662. Springer, Heidelberg (2009)
13. Bottou, L.: Stochastic Learning. In: Bousquet, O., von Luxburg, U., Rätsch, G. (eds.) Machine Learning 2003. LNCS (LNAI), vol. 3176, pp. 146–168. Springer, Heidelberg (2004)
14. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: Bm25 and beyond. Found. Trends Inf. Retr. 3, 333–389 (2009)
15. Bespalov, D., Bai, B., Qi, Y., Shokoufandeh, A.: Sentiment classification based on supervised latent n-gram analysis. In: ACM Conference on Information and Knowledge Management, CIKM (2011)
16. Lebanon, G., Mao, Y., Dillon, J.: The locally weighted bag of words framework for document representation. J. Mach. Learn. Res. 8, 2405–2441 (2007)
17. Bottou, L.E., Cun, Y.L.: Large scale online learning. In: NIPS 2003. MIT Press (2004)
18. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: International Conference on Machine Learning, ICML (2008)
19. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of The American Society for Information Science 41(6), 391–407 (1990)
20. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 50–57. ACM Press, New York (1999)
21. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. The Journal of Machine Learning Research 3, 993–1022 (2003)
22. Weston, J., Bengio, S., Usunier, N.: Large scale image annotation: learning to rank with joint word-image embeddings. Machine learning 81(1), 21–35 (2010)
23. Bengio, Y.: Learning Deep Architectures for AI. Now Publishers Inc., Hanover (2009)
24. Glorot, X., Bordes, A., Bengio, Y.: Domain adaptation for large-scale sentiment classification: A deep learning approach. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011). Omnipress, Bellevue (June 2011)
25. Socher, R., Lin, C.C.Y., Ng, A., Manning, C.: Parsing natural scenes and natural language with recursive neural networks. In: Getoor, L., Scheffer, T. (eds.) Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 129–136. ACM, New York (June 2011)
26. Bengio, Y., Ducharme, R., Vincent, P., Operationnelle, D.D.E.R.: A neural probabilistic language model. Journal of Machine Learning Research 3, 1137–1155 (2000)
27. Morin, F.: Hierarchical probabilistic neural network language model. In: AISTATS 2005, pp. 246–252 (2005)
28. Leslie, C.S., Eskin, E., Weston, J., Noble, W.S.: Mismatch string kernels for SVM protein classification. In: NIPS, pp. 1417–1424 (2002)
29. Weston, J., Leslie, C., Ie, E., Zhou, D., Elisseeff, A., Noble, W.S.: Semi-supervised protein classification using cluster kernels. Bioinformatics 21(15), 3241–3247 (2005)
30. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. J. Mach. Learn. Res. 2, 419–444 (2002)
31. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In: ACL, pp. 187–205 (2007)

32. Lewis, D.D., Yang, Y., Rose, T.G., Li, F., Dietterich, G., Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361–397 (2004)
33. Deshpande, M., Karypis, G.: Evaluation of Techniques for Classifying Biological Sequences. In: Chen, M.-S., Yu, P.S., Liu, B. (eds.) PAKDD 2002. LNCS (LNAI), vol. 2336, pp. 417–431. Springer, Heidelberg (2002)
34. Duskin, O., Feitelson, D.G.: Distinguishing humans from robots in web search logs: preliminary results using query rates and intervals. In: Proceedings of the 2009 Workshop on Web Search Click Data. WSCD 2009, pp. 15–19. ACM, New York (2009)

The Bitvector Machine: A Fast and Robust Machine Learning Algorithm for Non-linear Problems

Stefan Edelkamp and Martin Stommel

Research Group Artificial Intelligence, Universität Bremen
Am Fallturm 1, 28359 Bremen, Germany
{edelkamp,mstommel}@tzi.de

Abstract. In this paper we present and evaluate a simple but effective machine learning algorithm that we call *Bitvector Machine*: Feature vectors are partitioned along component-wise quantiles and converted into bitvectors that are learned. It is shown that the method is efficient in both training and classification. The effectiveness of the method is analysed theoretically for best and worst-case scenarios. Experiments on high-dimensional synthetic and real world data show a huge speed boost compared to Support Vector Machines with RBF kernel. By tabulating kernel functions, computing medians in linear-time, and exploiting modern processor technology for advanced bitvector operations, we achieve a speed-up of 32 for classification and 48 for kernel evaluation compared to the popular LIBSVM. Although the method does not generally outperform a SVM with RBF kernel it achieves a high classification accuracy and has qualitative advantages over the linear classifier.

Keywords: classification, support vector machine, time/accuracy tradeoff.

1 Introduction

Due to the flexibility of kernel functions, statistical machine learning with Support Vector Machines, SVMs for short [1], is one of the most successful approaches for classification applications [2].

The number of support vectors learned affects training and classification speed. Bordes et al. ([3]) assume that not all training samples are equally relevant for the resulting model. Their LASVM[1] implementation achieves a significant speedup during training by using an online approach, where the margin is determined from a small subset of the input data. Important modeling decisions can be reached even without taking into account the class label of a data point.

The measurement of the model accuracy also affects the training speed. Joachims' [4] SVMLIGHT[2] implementation achieves a significant speed-up in

[1] http://leon.bottou.org/projects/lasvm
[2] http://svmlight.joachims.org/

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 175–190, 2012.

the estimation of the leave-one-out error by evaluating the Lagrange multipliers and slack variables of a trained model. This saves the retraining of the model for every sample left out. Tsang et al. [5] propose a radical simplification of the model by approximating the convex hull of the training data by an enclosing ball in the high-dimensional feature space. The allowed error of the approximation is a parameter of the algorithm. The method works iteratively to find the centre of the ball. The idea of using a reduced set of support and also non-support vectors to define the decision border has also been used in previous approaches [6,7]. Influences stem for example from approximate k-nearest neighbours [8] or computational geometry [7].

There are also fast SVM implementations with linear kernels that exceed the library used in this paper (LIBSVM[3]) by far, e.g. LIBLINEAR[4] and Leon Bottou's stochastic gradient descent SVM[5]. However, for complex data where single classes comprise multiple distant sub-clusters, non-linear kernels achieve a higher accuracy. Chang et al. [9] achieve a speed-up of a second order polynomial classifier by using optimisation techniques usually reserved for linear kernels. Zhang et al. [10] compute a low rank linear approximation of the kernel matrix of a non-linear kernel and evaluate it by a linear SVM.

The dimensionality of the input data affects the speed of the classification as well, but it also often causes a numerical instability known as the *curse of dimensionality*. It is described [11] as a general unreliability of distance computations for data sets where minimum and maximum distances approximate with rising dimensionality. The effect can be reduced by choosing a smaller norm than the Euclidian [12], but taking high roots is numerically difficult, too.

In the application of machine learning algorithms to image and video data, a binary discretisation of the popular SIFT (Scale Invariant Feature Transformation [13]) feature vector does not suffer from this effect [14], whereas the original SIFT representation does. Although a feature binarisation is a dramatic simplification of the input data, it has been observed for SIFT and SURF (Speeded Up Robust Features [15]) descriptors that the matching accuracy does not decrease significantly [16]. In some cases the relative error rates even decreased. Moreover, the length of the descriptors is reduced by a factor of 8 for SIFT (128 dimensions, usually implemented as single bytes) and by a factor of 32 for SURF (64 dimensions, usually implementated by 4 byte floating point values). These advantages are highly relevant for robotics applications as SURF or SIFT are called at a high frequency in Simultaneous Localisation and Mapping (SLAM)[6].

In this paper, we study the findings on feature binarisation in a more general setting. A brief summary of SVMs allows us to introduce the method under the

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvm

[4] http://www.csie.ntu.edu.tw/~cjlin/liblinear

[5] http://leon.bottou.org/projects/sgd (this algorithm has been ported by us to the GPU for an even faster evaluation time)

[6] http://openslam.org

notion of a *Bitvector Machine*[7]. We discuss conditions under which the binarisation of a feature vector is appropriate. We observe that the Hamming distances between binarised input vectors are discrete and limited. This allows a number of important code optimisations for kernel evaluation, most notably the use of look-up tables and native processor instructions. In contrast to previous work [16] where no dedicated CPU instructions are used, we can therefore quantify the main advantage of the method: the speed in the kernel evaluation. For transforming the input data from floating point to binary strings we exploit that medians can be computed efficiently. By replacing the medians by q-quantiles, we generalise the Bitvector Machine to a Multi-Bitvector Machine. The method allows for a wider range of possible time-accuracy trade-offs and avoids some worst-case behaviours of the single-bit binarisation. In the empirical part of the paper we evaluate the (single-bit) approach in three different experimental settings. The first and second ones comprise artificial data (a Mixed Gaussian Distribution). The third one deals with real-world data from a face recognition application. As a result we achieve a several fold speed-up in the classification with only a small loss in accuracy compared to LIBSVM. As far as pure kernel computations are concerned, the method also accelerates training, although this is not deepened in the paper.

2 Support Vector Machines

Raw data presented to a supervised statistical machine learning algorithm [17] can be arbitrarily complex and is often mapped to a set of numerical values, called the feature vector. The classification problem deals with the prediction of the label l of previously unknown feature vectors $\mathbf{x} \in \mathbb{R}^d$ that constitute the test data. During training, a partitioning of the feature space $\mathbb{R}^d$ is learned, where each partition is assigned a label l from a small set L based on a set of training samples $(\mathbf{x}_1, l_1), \ldots, (\mathbf{x}_k, l_k) \in \mathbb{R}^d \times L$ with known labels. The challenge is to approximate the unknown distribution without overfitting the training data.

Support Vector Machines. [1], SVMs for short, achieve this task by learning coefficients for a kernel mapping to a high-dimensional space, where a linear class border is spanned up by a number of support vectors that outline the data. We keep the presentation brief as there are text books on SVMs and related kernel methods [2,18]. Theoretically, it should be sufficient to determine the class border by just three support vectors. However, it is not known in advance if any of the known kernels realises a suitable mapping. The use of generic kernels instead leads to a much larger number of support vectors (which critically influence classification time). In the worst case finding a separating hyperplane takes quadratic time in the number of data points.

The classification rule for a two-class non-linear classification function ϕ is

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{s} \beta_i(\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})) + b\right), \tag{1}$$

[7] The term machine links to the fact that the classifier realises a mathematical function, sometimes referred to as a machine.

where $\mathbf{x}_i$ is the ith of $s \leq k$ support vectors, β_i is a coefficient that includes class label and Lagrange multiplier from the optimisation, and b is some additional translation constant. Assuming a kernel function $K(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u}) \cdot \phi(\mathbf{v})$ we get

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{s} \beta_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b\right), \tag{2}$$

which does not refer directly to ϕ, known as the kernel trick.

SVM training is a convex optimisation problem which scales with the training set size rather than the feature space dimension. While this is usually considered to be a desired quality, in large scale problems it may cause training to be impractical and classification to be time consuming.

The corpus of SVM applications is large and encompasses many areas of computer science [2]. As kernel functions can be complex, the application of SVMs is large and kernels can be designed to cover simple regression to neural network approaches [19] and time series [20]. However, linear kernels $K(\mathbf{u}, \mathbf{v}) = \mathbf{u} \cdot \mathbf{v} + g$ or Gaussian (RBF) kernels $K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma \|\mathbf{u} - \mathbf{v}\|^2)$, $\gamma \in \mathbb{R}$ are the ones that are most commonly used. In the second case, a SVM is a function that itself defines a Mixed Gaussian Distribution [21] and is related to Radial Basis Networks and Neuronal Nets with one hidden layer. Since the optimisation objective is high accuracy instead of a low number of support vectors, a Mixed Gaussian input will be usually modeled by multiple centres per Gaussian in the SVM.

The running time for classifying one vector is $O(sde)$, where e is the time to evaluate the exponential. Practical SVM implementations might assume the input data to be normalised to avoid numerical difficulties.

3 Bitvector Machine

A *Bitvector Machine*, BVM for short, is an SVM with a binarisation (Boolean discretisation) of the input: All vectors $\mathbf{x}_i \in \mathbb{R}^d$, $1 \leq i \leq k$, used in the training phase and all vectors evaluated in the test phase are mapped to $\{0,1\}^d$. The labels remain unchanged. The results are then fed into a SVM.

The *median* of k totally ordered elements is the element in the $\lfloor k/2 \rfloor$-th position after sorting. Median selection can be performed in linear time [22]. Let $\bar{\mathbf{x}} = (\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^d)^\top \in \mathbb{R}^d$ be the component-wise median of the input vector, i.e. $\bar{x}^j$ is the median of the j-th vector components of $\mathbf{x}_1, \ldots, \mathbf{x}_k$.

The BVM maps (training and test) vectors $\mathbf{x} \in \mathbb{R}^d$ to binary strings $\mathbf{z} = (z^1, z^2, \ldots, z^d)^\top \in \{0,1\}^d$ as follows: For each j, $1 \leq j \leq d$, we have $z^j = 0$ if and only if $x^j < \bar{x}^j$. Otherwise z^j is set to 1.

Theorem 1 (Time Complexity Single-Bit Binarisation). *The binarisation of all $\mathbf{x}_i \in \mathbb{R}^d$, $1 \leq i \leq k$, takes time $\Theta(kd)$.*

Proof. Computing all component-wise medians of vectors $\mathbf{x}_i \in \mathbb{R}^d$, $1 \leq i \leq k$, i.e. $\bar{\mathbf{x}}$, requires $O(kd)$ time. Thresholding all vectors $\mathbf{x}_i \in \mathbb{R}^d$ component-wise with $\bar{\mathbf{x}}$ can be executed in time $O(kd)$. Considering the input size of the data, the time $O(kd)$ is optimal.

By using q quantiles instead of the medians (where $q = 2$) we can create more detailed feature representations of longer word length. Quantiles represent a subdivision of the domain of a random variable into q consecutive partitions of equal cumulative density $1/q$. The respective thresholds can be read from the cumulative distribution function or by recursively computing medians (if q is a power of 2). The assignment of a random value to its η-th, $1 \leq \eta \leq q$, quantile can be done in log-time by arranging the thresholds in a balanced binary tree. The representation of the index η of a quantile requires $m = \lceil \lg q \rceil$ bits.

A *Multi-Bitvector Machine*, MBVM for short, maps (training and test) vectors $\mathbf{x} \in \mathbb{R}^d$ to binary strings $\mathbf{z} = (z^1, z^2, \ldots, z^{md})^\top \in \{0,1\}^{md}$, $m = \lceil \lg q \rceil$. In order to binarise a d-dimensional feature vector, we compute quantiles of fixed q independently for every dimension. The binary feature vector is the concatenation of all d quantile indices η, the resulting length is thus md bits. Computing all component-wise quantiles of vectors $\mathbf{x}_i \in \mathbb{R}^d$, $1 \leq i \leq k$, requires $O(mkd)$ time, as computing the list of all quantiles of a k element set takes time $O(mk)$ (see Ex. 9.3-6, p. 223 in [23]). As $q < k$ and, subsequently, $m < \lceil \lg k \rceil$ this approach is still faster than sorting with its time complexity of $\Omega(k \lg k)$.

Corollary 1 (Time Complexity Multi-Bit Binarisation). *The multi-bit binarisation of all* $\mathbf{x}_i \in \mathbb{R}^d$, $1 \leq i \leq k$, *using* q*-quantiles takes time* $O(mkd)$, *with* $m = \lceil \lg q \rceil$.

Instead of reducing the dimension as done in related work, e.g. on random projections [24], we enlarge it. The motivation is that the increase in dimensionality is compensated by the speed-up by the lower bit-rate.

The computation of median (quantile) based binary representation corresponds to a partitioning and re-labeling of the feature space. If we were to split the data iteratively, we would build a kd-tree in $O(n \lg n)$ time [25], for which rectangular range queries take $O(\sqrt{n} + k)$ and membership queries take $O(\lg n)$ time [26]. In contrast, in the BVM the median splits are chosen independently of each other, one in each vector component. The binarisation therefore defines a partitioning of the feature space into regions, where all separating hyperplanes intersect in one point. Geometrically, it can be interpreted as moving the origin of the Cartesian coordinate system to $\bar{\mathbf{x}}$ and representing each resulting orthant by a bitvector $\{0,1\}^d$ that indicates the position relative to the iso-oriented hyperplanes. The bitvectors correspond to nodes in a d dimensional hypercube (an edge in the hypercube has Hamming Distance 1).

Let ψ be the mapping that performs the binarisation. We have

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{t} \delta_i \cdot K(\psi(\mathbf{x}_i), \psi(\mathbf{x})) + g\right). \tag{3}$$

Due to a different training, the number of support vectors t, weights δ and bias g might be different from the original values (s, β, b in Eq.2). Moreover, as $K(\psi(\mathbf{u}), \psi(\mathbf{v})) = \phi(\psi(\mathbf{u})) \cdot \phi(\psi(\mathbf{v}))$, we see that we are actually dealing with a different kernel that transforms data via $\phi \circ \psi$ from $\mathbb{R}^d$ first into the Boolean space $\{0,1\}^d$ before lifting it into higher dimensions.

Because of the symmetry property, distance metrics based on an element by element comparison (like Euclidean or Hamming distance) in $\{0,1\}^d$ yield only $d+1$ different values. All possible results of the kernel $K(\psi(\mathbf{x}_i), \psi(\mathbf{x}))$ can therefore be precomputed.

For the Gaussian kernel $K(\psi(\mathbf{u}), \psi(\mathbf{v})) = \exp(-\gamma \|\psi(\mathbf{u}) - \psi(\mathbf{v})\|^2)$ the term $\|\psi(\mathbf{u}) - \psi(\mathbf{v})\|^2$ can only yield $d+1$ different values. For the Euclidean norm they range from 0 to d and equal the Hamming distance of the kernel arguments. By applying the parameter γ and the exponential to these values, the whole kernel can be precomputed and stored in a table. This avoids the repeated time consuming computation of the exponential during classification. The Hamming distance can be computed by applying the population count instruction (counting the number of bits set) to the bitwise XOR disjunction of the arguments.

Precomputing kernels reduces training and classification time. Because training is done only once, we focus on the latter.

Theorem 2 (Time Complexity Single-Bit Classification). *Assuming d to be $O(w)$ for the computer word width w and native population count, the running time for classifying one bitvector is $O(t+d)$, where t is the number of support vectors.*

If population count is not native on the word level, then the classification of one vector has the complexity $O(d+t \lg^* d)$, where $\lg^* d$ is the iterated logarithm, i.e. the height of the shortest tower of powers $2^{2^{\cdots}}$ that equals or exceeds d.

Proof. Computing the binarisation $\psi(\mathbf{x})$ of the test vector $\mathbf{x}$ takes time $O(d)$. The population count and XOR to be executed on the word level to compute the Hamming Distance run in $O(1)$. Given that the kernel is tabulated, we require only lookups to the kernel table, so that multiplication with a constant and addition have to be executed t times to evaluate the classification formula $\sum_{i=1}^{t} \delta_i \cdot K(\psi(\mathbf{x}_i), \psi(\mathbf{x}))$.

For larger values of d population count can be done in $O(\lg^* d)$ by iterating the HAKMEM algorithm[8].

The second part of the theorem assumes large word width w and is mainly of theoretical interest.

Assuming that the binary representation of the bitvectors due to computing the quantiles still fits into a computer word width w, for an MBVM computing the binarisation of the test vector $\mathbf{x}$ takes time $O(md)$. Hence, the running time for classification generalizes to $O(t+md)$ with m being the dual logarithm of the number of quantiles considered.

Corollary 2 (Time Complexity Multi-Bit Classification). *Assuming d to be $O(w)$ for the computer word width w and native population count, the running time for classifying one bitvector is $O(t+md)$, where t is the number of support vectors.*

If population count is not native on the word level, then the classification of one vector has the complexity $O(md + t \lg^*(md))$.

[8] David Eppstein, http://11011110.livejournal.com/38861.html

The entropy for one split along the median is certainly maximal as long as class labels are not taken into account. But even for simplified Mixed Gaussian Distributions (2D, shifted mean, same deviation but same amplitude) entropies can only be approximated [27].

The number of regions distinguishable by the BVM rises exponentially with dimensionality. For dimensionality d we have 2^d possible regions. As we split the data component-wise, the BVM corresponds to a static decision tree that has depth d and that is independent of the number of elements. An explicit construction of such a tree, however, is not required. In contrast to SVMs, the binarisation automatically normalises the input data to the unit hypercube.

Because the BVM is trained the same way as a SVM, we can still use cross-validation or leave-one-out-validation to estimate the classification error.

4 Case Studies

One question is if and when the binary kernel is better than a linear one. If we assign each vector $\{0,1\}^n$ with even population count with class 1 and each vector $\{0,1\}^n$ with odd population count with class 2, then we generalise the XOR problem to higher dimension (the minimal Hamming distance of two elements class in one class is two). This is clearly not linearly separable, but the BVM can find a perfect classification.

This clearly is a best-case scenario, but we can argue that linearly non-separable but binary separable examples are common in practice. One reason for this is that many classes underly the principles of differentiation and composition. Let us assume for example a set of images of noses, either taken from a left angle or from a frontal perspective. Although left noses are visually and numerically similar to each other, they differ strongly from frontal noses and occupy a separate region in feature space. The combined class *nose*, however, is activated by features from both differentiations. The dispersed placement of multiple clusters of sub-classes in feature space can easily create non-linearly separable situations, especially in multi-class problems. Moreover, real-world data often comes from independent or principle component analyses. As a result, feature distributions for different sub-classes tend to be aligned with the coordinate axes.

One worst-case scenario for the BVM in two dimensions is a checker-board layout of two classes because after two orthogonal cuts all fields of the checker-board that fall into the same quadrant are represented by the same bitvector and with it the same class. For this theoretical setting, the SVM calls for several support vectors and likely an overfitting of the data. If the number of Gaussian kernels is small and the dimension is high, the BVM has good chances to find a discriminative partitioning. As a result, feature binarisation preserves high selectivity and lifts the curse of dimensionality.

However, if we were to use a MBVM with as many quantiles as the checker-board size, the worst-case behavior would be avoided and we would encounter a best-case scenario with accurate class boundaries.

Even though entire orthants collapse to single data points on the hypercube, the Gaussian weighting of the binary vectors still preserves at least some geometrical meaning. The weights for a specific class are propagated nonlinearly from one orthant to another via shared hyperedges. For shared hyperedges the Hamming distance of the associated bitvectors is one.

5 Experiments

We performed the experiments on one core of a desktop computer (model Intel Core i7 920 CPU 2.67 GHz) running Ubuntu 10.10 (Linux kernel 2.6.32-23-generic) with 24 GB main memory. With such memory capacity, there was no need to use virtual memory. We compiled all programs using GNU C++ compiler (`gcc` version 4.3 with option `-O3` and `-mpopcnt`).

For the implementation of the BVM we have extended LIBSVM to support bitvector manipulation based on a precomputed kernel and native population counting. LIBSVM is chosen because it is well known, widely used, and easily extendible. It uses a one-versus-one strategy for multi-class problems.

The use of SVM implementations aiming at higher training speed [3,4] would not provide a deeper insight because our method aims at testing speed. The BVM is also not in contradiction to approximative methods working on reduced sets [5,6,7,8]. It would therefore be possible to combine a fast binary vector representation with a small approximated set of support vectors in order to achieve an even higher speed-up. However, in favour of the clarity of the presentation we refrain from such combined approaches.

We evaluate the BVM in two synthetic and one natural scenarios.

5.1 Artificial Data

To validate our ideas experimentally, we produced training and test data for two and five-class problems in a random process with defined statistical properties. For each class we realised a sampling of a Mixture of Gaussians. The mean $\mu_i, i = 1, 2, \ldots$ of each multivariate Gaussian

$$p_i(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma^{-1}(\mathbf{x}-\mu_i)\right) \tag{4}$$

is placed in the unit hypercube. The bandwidth is set globally in the main diagonal of the covariance matrix Σ for all Gaussians. The number of Gaussians is set individually for every experiment.

The maximum likelihood estimate (Bayes) of the Mixed Gaussian distribution is used as the ground truth to which an SVM and the proposed method are compared. Linear kernels are used to detect linearly separable situations.

The difficulty of the created problems is controlled by adjusting the bandwidth of the Gaussians for a specific accuracy of the maximum likelihood estimator. If we keep the number of Gaussians fixed and increase the dimensionality, we have to increase the bandwidth, too. Otherwise the overlap between the

Gaussians decreases and the difficulty of the problem changes. The bandwidth of the kernel is therefore a parameter of the experimental setting. It exhibits a strong influence on the resulting estimate. We therefore set it manually, although Silverman indicates a dependency of the bandwidth on the root of the dimensionality [28].

First Scenario. As a parameter of the synthetic experiment, we choose the bandwidth to provide accuracy values to be in the range $[0.8, 0.85]$. For determining the classification accuracy (on the training and test sets) we conducted two experiments, one for a two-class problem and one for a five-class problem both with rising dimension (dimensionalities 2, 4, 8, ..., 256). The two-class problem is modeled by three Gaussians per class at uniformly distributed random positions in the unit cube. With 5 Gaussians per class, the five-class problem is more complex. Each Gaussian is sampled 70 times. The data set is randomised and split into equally sized training and test sets. The parameters C (a weighting of the slack variables in the optimised function) and γ of both the BVM and the RBF kernel of LIBSVM are optimised in a grid search using the Python scripts provided by LIBSVM. For the Gaussian kernel and the BVM the training set is further subdivided in order to perform a five-fold cross validation for parameter selection.

The plots in Fig. 1 show that the BVM solves the problem surprisingly well, given the limitation that the BVM does not represent any gradual or continuous feature values.

For smaller dimensions and the two-class problem, all methods lead to very similar results with a small advantage for the SVM with Gaussian kernel. The results of the BVM are comparable to those of the linear classifier, with changing winners. For the five-class problem, the results are clearer. Training and test results are closer for smaller dimensionalities and the BVM achieves results between the SVM with Gaussian and linear kernel, although closer to the linear one. The recognition rate of the BVM has a maximum at 32 dimensions. It seems that lower dimensions limit the capacity to represent information.

For more than 32 dimensions, the accuracy drops significantly for all methods. The pronounced divergence between the training and test results of the linear classifier indicates heavy overfitting. The Gaussian kernel does not show this overfitting but the bad results indicate a clear failure, too. Although the cross-validation on subsamples of the training set leads to better estimates of the final recognition rate, it cannot compensate for missing information.

In summary the accuracy of all classifiers is limited, so our experiment may be overly complex. Even the linear kernel might be competitive if additional application constraints are taken into account. It must also be said that the difficult class borders in this Gaussian mixture do not have much in common with data sets from pattern recognition tasks: In such applications, the feature vector often represents the similarity of a query object to a set of prototypes, where each similarity is stored in one dimension. Consequently, the class border would not normally cross a coordinate axis multiple times. The effect is even reinforced by the common use of coordinate transforms such as principle component analysis.

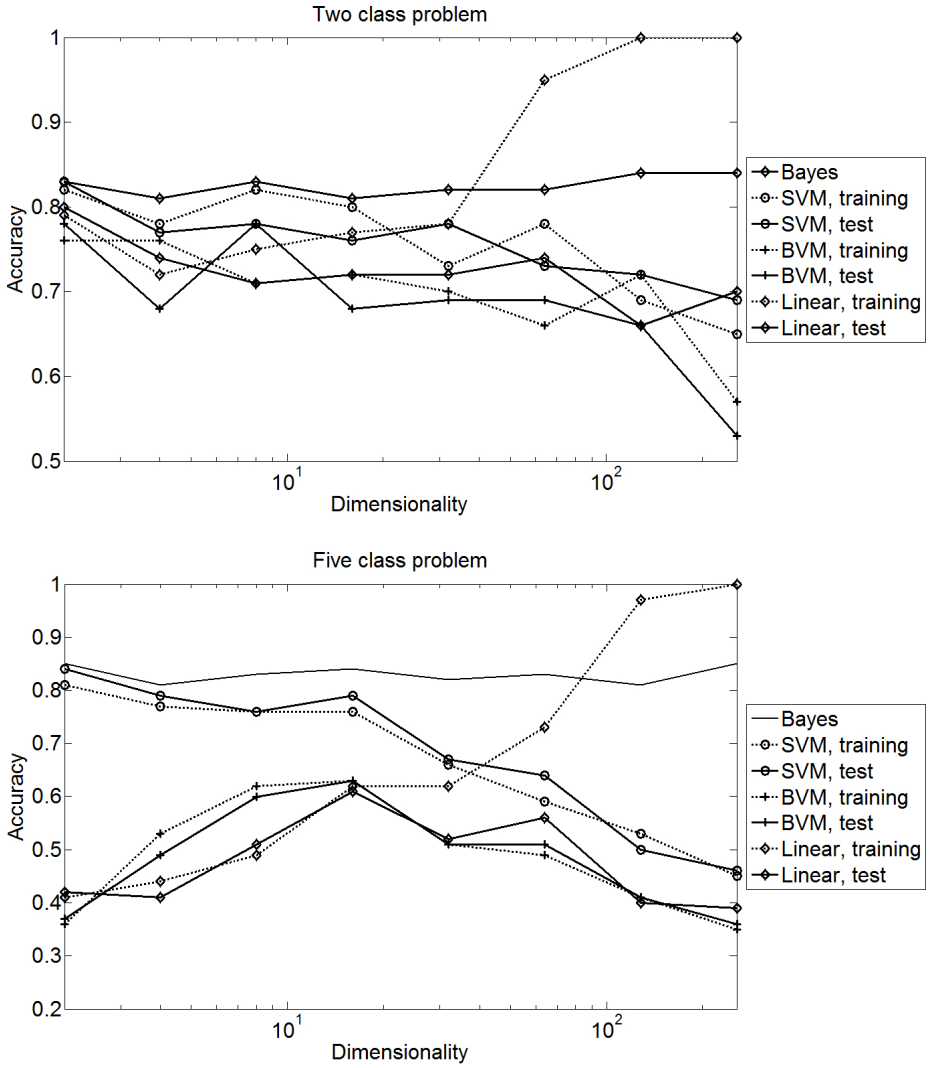


Fig. 1. First scenario: Accuracy for the SVM (RBF and linear kernel) and the BVM

In matters of CPU time we see a drastic decrease in computation time for the BVM. Figure 2 shows that we obtain a huge speed-up that additionally increases with the dimension of the problem. The maximum increase of performance was a factor of 32 for the two class problem in 128 dimensions. The increase in CPU time at $d = 256$ for the BVM can be explained by the increased word length. Here all word level operations have to be executed four times on a 64 bit machine. For bigger dimensions, computer architecture and compiler must be taken stronger into account. We have not measured the time for the preprocessing step.

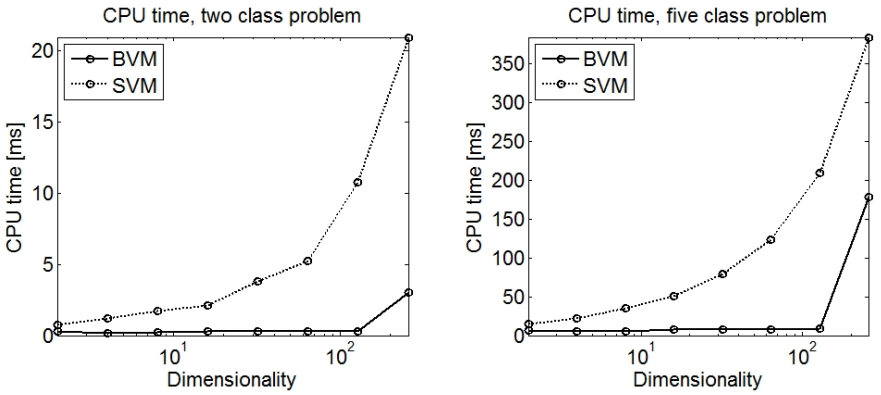


Fig. 2. CPU time measured for the classification of the data sets in the first scenario. The original real valued data is classified by an SVM with Gaussian kernel. The binary data refers to the BVM.

However, it seems obvious to us that a loop over d dimensions in the binarisation of a feature descriptor is uncomparably faster than the computation of dot products to thousands of d-dimensional support vectors in the classification of one descriptor.

The Second Scenario represents the above-mentioned high-dimensional XOR problem but with noisy data. To this end, we centre the Gaussians of our Gaussian Mixture at the corners of the unit hypercube and assign class labels to the Gaussians as described above. The bandwidth is adjusted so that the Bayes classifier achieves an accuracy of 90%–92% using the known distribution.

Figure 3 shows the results of the proposed method and the SVM using the RBF and linear kernel. The Gaussian Mixture consists of four planar XOR problems placed at random sides of the unit hypercube of dimensionality 8, 16, ..., 256). A planar XOR problem consists of four Gaussians at the corners of a square with diagonally different class labels. Each Gaussian is sampled 70 times. The best results can be seen for the SVM using the RBF kernel. The output of the BVM follows the SVM at a lower level. The linear classifier fails completely because it cannot represent the non-linear class borders.

Figure 4 shows the results for an 8-dimensional XOR arrangement of varying sparseness. For this distribution, Gaussians are randomly placed in a certain percentage of all corners and sampled 100 times each. For a sparse filling of 10%, the distribution still seems linearly separable. However, with increasing density the linear classifier quickly approaches random, whereas the BVM improves gradually and finally outperforms the SVM with RBF kernel (corner fill factor ≥ 0.5).

This good result shows that the interpretation of the BVM as a simplified approximation of the SVM depends on the understanding of the data (and studying the data is always a good start). If it is not known how the data corresponds

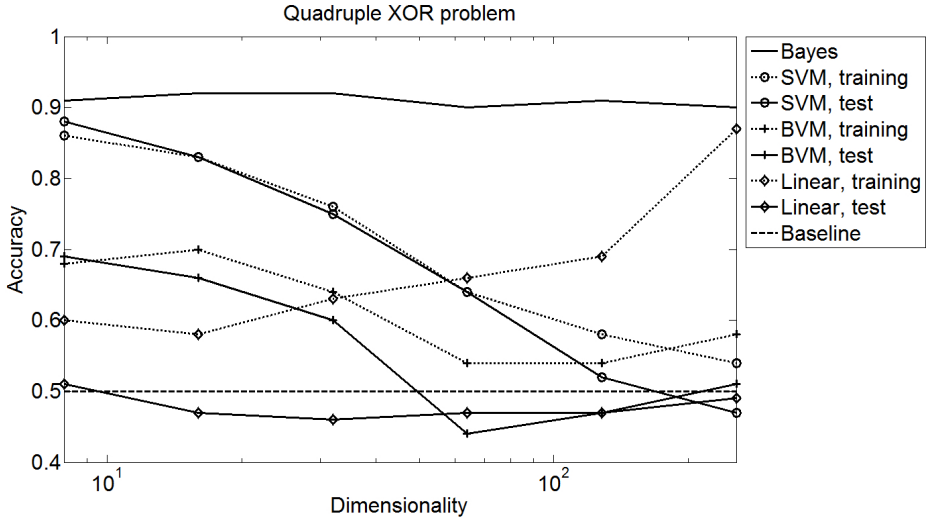


Fig. 3. Second scenario: Accuracy for a two class problem including four planar xor problems

to the described case scenarious, then the BVM must be considered as a non-parametric, simplified method and the results can be inferior to the SVM. If it is known that the data corresponds to a good case scenario, then we use the BVM as a model based method that benefits from our prior knowledge about the data.

5.2 Real World Data

The Third Scenario is a Computer Vision task, where SIFT descriptors [13] are classified into 16 classes (15 classes representing different parts of a face plus one background class). Additionally to the original SIFT method, we also tested a variation that takes into account that faces are shown in an upright orientation in most images. Without going into detail, we can say that the original SIFT method is rotationally invariant, i.e. different rotated versions of the image result in approximately the same feature vector. Our variation in contrast (marked as 'absolutely oriented' in Tab. 1) is selective to orientation, so different rotated versions of the same visual pattern can be distinguished. In both cases the binarised feature vectors have 128 bits, so they can be compactly stored in two 64-bit words.

We evaluate the effectiveness of population counting on the machine and study the speed-ups obtained for sole kernel evaluations (Table 2) and whole vector classifications (Table 3). We compare the LIBSVM implementation with the BVM in two settings, one with a precomputed 16-bit population count lookup-table (2^{16} entries), one with the native population count (`__builtin_popcountll`). We run three examples for each setting to show that the variance in the running times is

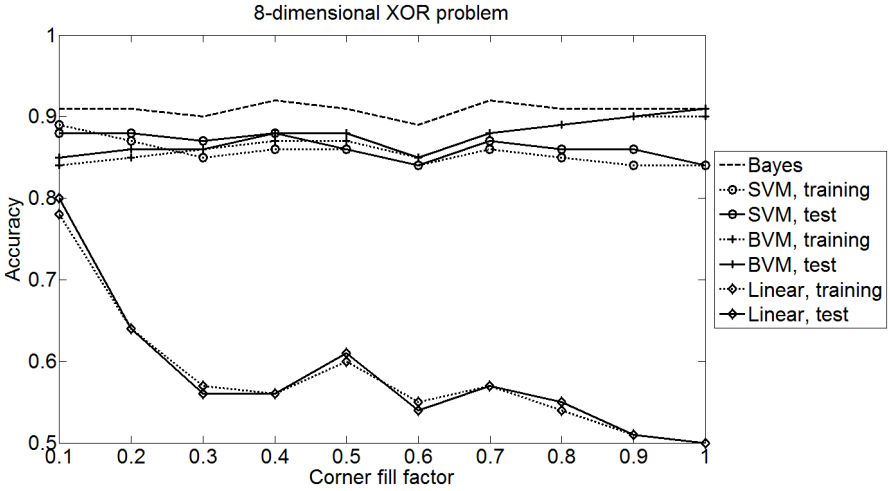


Fig. 4. Accuracy for a non-linear, 8-dimensional class arrangement of increasing complexity

small. The table documents a speed-up factor of 48 in the 116 million kernel comparisons and an improvement by a factor of 17 for the entire classification process. Further speed-ups might be achieved by using a one-versus-all strategy for multi-class problems instead of the one-versus-one strategy implemented in LIBSVM.

The difference in CPU time between the sole kernel computation and the whole classification indicates a strong influence of the code analysis and generation of the compiler, since the number of kernel computations has been equal in both experiments, and the optimisation flags too.

Table 1 shows for different SIFT variations and different class distributions that despite the high speed-up the accuracy of the BVM is on average (from the cross validation) close to that of the SVM. The remaining gap seems statistically significant when comparing the results of the training and test sets. However, it becomes lower with increasing sample size. In comparison to the linear classifier, the BVM is clearly better. For the larger data sets, the BVM is closer to the

Table 1. Accuracy [%] for SIFT data sets of different size and class distribution. There are 15 foreground classes and one background class. The samples are randomised and split into equally sized training and test sets.

Feature vector	SIFT		SIFT, absolutely oriented					
Foreground samples per class	20		20		125		250	
Background samples	2000		2000		1125		2250	
Classifier	Training	Test	Training	Test	Training	Test	Training	Test
LibSVM, RBF	73.4	74.0	79.9	79.1	84.7	85.6	86.8	87.9
BVM, RBF	67.2	67.3	74.0	75.0	82.4	83.2	84.2	85.1
LibSVM, linear	99.1	60.7	97.7	70.5	94.6	77.4	90.9	79.6

Table 2. Evaluation time [s] for 116 072 232 kernel computations. The first column gives the result for LIBSVM using the Gaussian kernel. The second column gives the results for the BVM, where a look-up-table for 16-bit wide sub-words of the feature vector is used for the computation of the population count. In the last column, the native 64-bit population count CPU instruction is used.

No	LIBSVM	BVM, 16-bit-popcnt	BVM, 64-bit popcnt
1	48.36	2.16	1.00
2	48.27	2.16	1.00
3	48.59	2.14	1.01

Table 3. Time [s] for classifying 16 788 vectors in 16 classes, 6914 support vectors

	Kernel computation using all support vectors		
No	LIBSVM	BVM, 16-bit popcnt	BVM, 64-bit popcnt
1	39.44	4.01	2.84
2	39.45	4.01	2.81
3	39.71	4.00	2.71
	only support vectors with $\beta \neq 0$		
1		3.46	2.27
2		3.45	2.28
3		3.46	2.28

Gaussian SVM than the linear one. The good results match findings in [16] where the classification accuracy (measured in error rates) has been shown to behave well for other Computer Vision Tasks, too.

6 Conclusion

We proposed a machine learning algorithm whose advantages result from a dramatic simplification of the input data. Discretisation certainly has limits in the accuracy of class distributions that are not iso-oriented. We argue, however, that iso-oriented classes with non-linearly separable sub-classes occur in many pattern recognition tasks.

The rising number of positive results in discretising feature vectors into bitvectors prior to the learning process shows that the effectiveness and efficiency of the binarisation in the BVM is an exciting phenomenon. Especially for a growing number of dimensions, where we lack a visual interpretation and where unexpected results like the curse of dimensionality have been measured, research might have concentrated on aspects that do not discriminate well. Even though binarisation reduces the information in the input considerably our experiments show that the results are often of acceptable quality, sometimes even better than the original unabstracted input.

Our experiments on synthetic and real data show that the accuracy of the BVM approximates (and in special cases exceeds) the accuracy of a SVM with RBF kernel better than a linear classifier, but is up to 48 times faster in the

kernel computation and up to 32 times faster in classification. These results confirm our theoretical proof of the efficiency of classification and binarisation. Compared to a linear classifier, the accuracy of the BVM is usually higher or equal. Furthermore, the BVM has the ability to model non-linearly separable problems where the linear classifier fails. Together with an improved empirical basis we provided insights that increase the understanding of when and why the approach works well, especially for large feature vectors. The BVM therefore allows for a welcome new trade-off between accuracy and running time for non-linear problems. The use of q-quantiles with $q > 2$ in the MBVM can improve the accuracy by the cost of time performance. Hence, the approach is expected to be applicable to problems that are not iso-oriented.

References

1. Vapnik, V.N., Chervonenkis, A.Y.: Theory of Pattern Recognition. Nauka, USSR (1974) (in Russian)
2. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press (2000)
3. Bordes, A., Ertekin, S., Weston, J., Bottou, L.: Fast Kernel Classifiers with Online and Active Learning. Journal of Machine Learning Research 6, 1579–1619 (2005)
4. Joachims, T.: Learning to Classify Text using Support Vector Machines. Kluwer (2002)
5. Tsang, I., Kocsor, A., Kwok, J.T.: Simpler core vector machines with enclosing balls. In: Ghahramani, Z. (ed.) 24th International Conference on Machine Learning (ICML), pp. 911–918. ACM, New York (2007)
6. Burges, C.J.C.: Simplified Support Vector Decision Rules. In: Proceedings of the Thirteenth International Conference on Machine Learning (ICML), pp. 71–77. Morgan Kaufmann (1996)
7. DeCoste, D.: Anytime Interval-Valued Outputs for Kernel Machines: Fast Support Vector Machine Classification via Distance Geometry. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 99–106 (2002)
8. Decoste, D., Mazzoni, D.: Fast query-optimized kernel machine classification via incremental approximate nearest support vectors. In: International Conference on Machine Learning (ICML), pp. 115–122 (2003)
9. Chang, Y.W., Hsieh, C.J., Chang, K.W., Ringgaard, M., Lin, C.J.: Training and Testing Low-degree Polynomial Data Mappings via Linear SVM. Journal of Machine Learning Research 11, 1471–1490 (2010)
10. Zhang, K., Lan, L., Wang, Z., Moerchen, F.: Scaling up Kernel SVM on Limited Resources: A Low-rank Linearization Approach. In: International Conference on Artificial Intelligence and Statistics, AISTATS (2012)
11. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When Is Nearest Neighbor Meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
12. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the Surprising Behavior of Distance Metrics in High Dimensional Space. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 420–434. Springer, Heidelberg (2000)
13. Lowe, D.G.: Object Recognition from Local Scale-Invariant Features. In: International Converence on Computer Vision (ICCV), pp. 1150–1157 (1999)

14. Stommel, M., Herzog, O.: Binarising SIFT-Descriptors to Reduce the Curse of Dimensionality in Histogram-Based Object Recognition. In: Ślęzak, D., Pal, S.K., Kang, B.-H., Gu, J., Kuroda, H., Kim, T.-h. (eds.) SIP 2009. CCIS, vol. 61, pp. 320–327. Springer, Heidelberg (2009)
15. Bay, H., Ess, A., Tuytelaars, T., Gool, L.J.V.: Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding 110(3), 346–359 (2008)
16. Stommel, M., Langer, M., Herzog, O., Kuhnert, K.D.: A Fast, Robust and Low Bit-Rate Representation for SIFT and SURF Features. In: Proc. IEEE International Symposium on Safety, Security, and Rescue Robotics, pp. 278–283 (2011)
17. Summa, M.G., Bottou, L., Goldfarb, B., Murtagh, F., Pardoux, C., Touati, M. (eds.): Statistical Learning and Data Science. CRC Computer Science & Data Analysis. Chapman & Hall (2011)
18. Schoelkopf, S.: Learning with Kernels. MIT Press (2001)
19. Schneegaß, D., Schäfer, A.M., Martinetz, T.: The Intrinsic Recurrent Support Vector Machine. In: European Symposium on Artificial Neural Networks (ESANN), pp. 325–330 (2007)
20. Gudmundsson, S., Runarsson, T.P., Sigurdsson, S.: Support vector machines and dynamic time warping for time series. In: International Joint Conference on Neural Networks (IJCNN), pp. 2772–2776 (2008)
21. Permuter, H., Francos, J., Jermyn, I.H.: A study of Gaussian mixture models of colour and texture features for image classification and segmentation. Pattern Recognition 39(4), 695–706 (2006)
22. Blum, M., Floyd, R., Pratt, V., Rivest, R., Tarjan, R.: Time bounds for selection. J. Comput. System Sci. 7, 448–461 (1973)
23. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press (2009)
24. Voloshynovskiy, S., Koval, O., Beekhof, F., Pun, T.: Random projections based item authentication. In: Proceedings of SPIE Photonics West, Electronic Imaging / Media Forensics and Security, vol. 7254 (2009)
25. Bentley, J.L.: Multidimensional Binary Search Trees Used for Associative Searching. Commun. ACM 18(9), 509–517 (1975)
26. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry Algorithms and Applications, 3rd edn. Springer, Heidelberg (2008)
27. Michalowicz, J.V., Nichols, J.M., Bucholtz, F.: Calculation of Differential Entropy for a Mixed Gaussian Distribution. Entropy 10(5), 200–206 (2008)
28. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)

Embedding Monte Carlo Search of Features in Tree-Based Ensemble Methods

Francis Maes, Pierre Geurts, and Louis Wehenkel

University of Liège
Dept. of Electrical Engineering and Computer Science
Institut Montefiore, B28, B-4000, Liège - Belgium

Abstract. Feature generation is the problem of automatically constructing good features for a given target learning problem. While most feature generation algorithms belong either to the *filter* or to the *wrapper* approach, this paper focuses on *embedded* feature generation. We propose a general scheme to embed feature generation in a wide range of tree-based learning algorithms, including single decision trees, random forests and tree boosting. It is based on the formalization of feature construction as a sequential decision making problem addressed by a tractable Monte Carlo search algorithm coupled with node splitting. This leads to fast algorithms that are applicable to large-scale problems. We empirically analyze the performances of these tree-based learners combined or not with the feature generation capability on several standard datasets.

Keywords: Embedded Feature Generation, Monte Carlo Search, Decision Trees, Random Forests, Tree Boosting.

1 Introduction

It is often admitted that the successful application of supervised learning depends at least as much on the features chosen to describe the inputs of objects than on the adopted learning algorithm. In addition to improving the accuracy of the resulting models, a proper choice of features can also lead to more compact models which often gain in interpretability. In practice, feature engineering - the process of identifying a good set of features for a given learning task - is usually performed manually based on problem expertise, which makes it more an art than a science. In order to remedy this situation, a number of algorithms for automatic feature generation have been proposed since the nineties (see [15,19] for examples of early work on this topic).

Most proposed approaches for automatic feature generation[1] take the form of a preprocessing: before doing actual learning, some kind of search is performed in a space of candidate features in order to construct a (typically small) set of features that are expected to help learning better models. Proposed approaches for this preprocessing can be classified in two categories: *filters* and *wrappers* [6]. In the

[1] This task is also known as automatic feature discovery, construction, or extraction.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 191–206, 2012.

former case, the search for good features is performed on the basis of general statistics, logical or information content criteria (see [8] for example), while the latter case directly relies on the performance of the target learning algorithm to guide search through the feature space. Examples of wrappers include the work of [16] based on wrapping the kNN learning algorithm or the work of [10] where the wrapped learning algorithm is a C4.5 decision tree. More recently the authors of [18] proposed an algorithm for joint feature construction and feature selection, wrapping either C4.5, kNN or a Bayesian classifier. Some form of genetic programming is used in most of these works, in which individuals are typically feature sets represented as a forest composed of n trees, each tree describing one particular feature.

Feature generation is closely related to feature selection and, while feature selection methods may also be classified as filters or wrappers, the last decade has seen an increasing interest for so-called *embedded* feature selection methods. In these latter methods, the feature selection task is embedded within the learning algorithm formulation, for example through the use of a L1-norm based regularization term added to an average loss term to yield the learning objective function [13]. Embedded methods may offer some advantages over filters and wrappers, including much better scaling properties and better theoretical understanding. Surprisingly however, embedded methods have received little attention in the field of feature generation. To our best knowledge, the few embedded feature generation methods proposed so far are built around single decision tree induction. As an example, it is proposed in [5] to invoke a genetic programming algorithm to find the best splitting feature at each node during decision tree induction. In this case, feature generation is not seen anymore as a preprocessing step, but is instead tightly integrated within the learning process.

In this paper, we propose a general scheme to embed in a flexible way feature generation in a wide range of tree-based supervised learning algorithms including single decision trees, random forests and common forms of tree boosting. We emphasize our analysis on the two latter types of algorithms, since numerous studies show that they clearly outperform single decision trees in terms of classification accuracy [2].

Both random forests and tree boosting rely on some form of vote over a set of predictors and they are often the most effective when the individual predictors are only of moderate quality: boosting is based on the combination of many weak classifiers and random forests rely on randomization to reduce the correlation of ensemble terms. As a consequence, it may be unnecessary and possibly counterproductive to invest a huge computational budget in the search over the feature space in the context of these methods. Therefore, instead of using computationally complex genetic programming algorithms, we propose to use a Monte Carlo search algorithm which budget may be controlled so as to both weakly and efficiently explore the feature space at each tree-node when constructing model terms in random forests or in tree boosting. Our approach thus leads to a fast integrated learning and feature generation procedure that scales well to large scale problems and adapts well to the properties of tree-based ensemble

methods, while it works also with single trees. We show empirically that embedding this feature search into single trees, random forests and tree boosting yields significant improvements over these algorithms in their basic forms.

The rest of this paper is organized as follows. Section 2 introduces notations and formulates the learning problem we address. Section 3 motivates the main principles of our approach for embedded feature generation. Section 4 formalizes feature generation as a sequential decision making problem and describes Monte Carlo search algorithms to explore the feature space both efficiently and with controlled strength. Section 5 presents an empirical evaluation of our algorithms using several tree-based learning algorithms and Section 6 concludes.

2 Problem Formulation

We consider supervised learning where, given a dataset $S = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i\in[1,N]}$ of samples $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ i.i.d. from a distribution $\mathcal{D}_{\mathcal{X},\mathcal{Y}}$, the aim is to infer a classifier $h \in \mathcal{H}$ to minimize the expected risk: $\mathsf{E}_{(\mathbf{x},y)\sim\mathcal{D}_{\mathcal{X},\mathcal{Y}}}\{\Delta(h(\mathbf{x}), y)\}$. Classically in supervised learning, the input objects $\mathbf{x}$ are vectors of numerical or categorical features that can directly be exploited by the learning algorithm. This assumes that feature engineering has already been done when formulating the learning problem. In our context, the aim is to integrate feature generation within the learning process and we thus make no strong assumptions on the nature of $\mathcal{X}$. An input object $\mathbf{x} \in \mathcal{X}$ is an n-tuple of properties $\mathbf{x} = (x_1, \ldots, x_n)$, where each x_i belongs to the space $\mathcal{X}_i$ that can either be continuous, discrete or structured. Properties can for example be raw signals such as images or structured data such as trees and graphs. Classical categorical or numerical data also naturally fit in this framework.

In order to bring the capacity of feature generation to the learning algorithm, we expect the user to provide a set of *constructor functions*, as proposed in [11]. These functions can be mathematical, logical and/or domain-specific and serve as the basis for feature generation. Formally, a constructor function of arity a is a triplet $(\mathcal{I}, \mathcal{O}, F)$ where $\mathcal{I}$ is the input domain $(\mathcal{X}_1 \times \cdots \times \mathcal{X}_k)$, $\mathcal{O}$ is the output domain and F is a function $F : \mathcal{I} \to \mathcal{O}$. As an example, *addition of two scalars* has arity $k = 2$ and is defined as $(\mathcal{I}, \mathcal{O}, F) = (\mathbb{R} \times \mathbb{R}, \mathbb{R}, F(x, y) = x + y)$. Constructor functions can either be applied to the input properties x_i or to the results of other constructor functions. This naturally leads to tree-structured features as illustrated by Figure 1. Note that this way of generating features is rather general and enables to encode complex processing pipelines [14].

3 A General Scheme for Embedding Feature Generation

Most previous automatic feature generation algorithms are preprocessings that aim at constructing a mapping $\phi : \mathcal{X} \to \mathcal{Z}$ that extracts a set of relevant features adapted to the targeted learning algorithm (e.g. typically, we have $\mathcal{Z} \subset \mathbb{R}^d$). Learning is then performed on the basis of the modified data set $\{\phi(\mathbf{x}^{(i)}), y^{(i)}\}_{i\in[1,N]}$ and classifying a new input $\mathbf{x} \in \mathbf{X}$ aims at computing

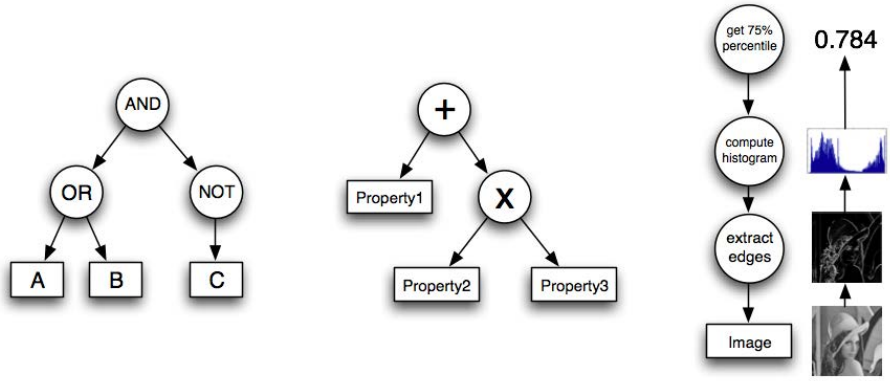


Fig. 1. Examples of constructed features for three different domains: booleans, real-valued attributes and images

$h(\phi(\mathbf{x})) \in \mathcal{Y}$. This paper proposes a scheme to do both feature generation and learning in an integrated way, so as to solve the following problem: given the dataset whose inputs are general properties, and a set of constructor functions, infer a classifier $h : \mathcal{X} \to \mathcal{Y}$ to minimize the expected risk.

Random forests, boosted stumps and boosted decision trees are among the top-performing and most widely used supervised learning methods nowadays [2]. These algorithms all rely on decision trees, which involve splitting the dataset recursively by testing one variable at a time. At this level, instead of testing the raw input variables of the problem, our approach consists in splitting the local dataset on the basis of locally constructed features, along the ideas of [5].

Table 1 summarizes the characteristics of the splitting procedure of well-known tree-based supervised learning algorithms[2]. In single decision trees and random forests, splits are constructed by searching the threshold optimizing the information gain for each candidate variable and by selecting the variable with maximal information gain. Random forests [1] are forests constructed using bagging and *random subspaces*: when building a node, only a subset of the (non-constant) attributes are considered as candidates for making the split. In addition to random subspaces, extremely randomized trees [7] introduce randomization by selecting splitting thresholds randomly, which often leads to improved models. In the two boosting methods, the splitting criterion depends on both the selected samples D and on their current weight W [17].

Our approach which is detailed in Section 4 consists in testing a small and randomized part of the feature space to *weakly* optimize the split scores of Table 1, during each node creation. Note that, except single decision trees, all these learning algorithms are ensemble approaches that rely on some form of majority vote to perform predictions. We believe that in this context, performing only a weak optimization over the feature space makes sense for various reasons:

[2] Note that while we only consider numerical attributes for the sake of simplicity, the ideas developed in this paper also apply to other types of attributes.

Table 1. Tree-based learning algorithms with associated splitting procedures

Method	Abbr.	Split score	Optimizer
Single decision Tree	ST	$\max_{t \in \mathcal{X}_i} InfoGain(S, x_i \leq t)$	Exhaustive
Random Forest	RF	$\max_{t \in \mathcal{X}_i} InfoGain(S, x_i \leq t)$	Subset
Extremely randomized Trees	ET	$InfoGain(S, x_i \leq t)$ where $t \sim \mathcal{U}_{\mathcal{X}_i}$	Subset
Boosted Stumps	BS	$\max_{t \in \mathcal{X}_i} Edge(S, W, x_i \leq t)$	Exhaustive
Boosted decision Trees	BT	$\max_{t \in \mathcal{X}_i} Edge(S, W, x_i \leq t)$	Exhaustive

- With all ensemble models, learning typically involves making several thousands of splits. So, even if only, say, 1% of a feature subspace is looked at each node, the whole subspace may still be visited multiple times during the entire learning procedure, and each of its elements may have multiple chances to be selected.
- Random forests and extremely randomized trees already rely on random subspaces to introduce randomization, which was shown to lead to improved generalization accuracy. Hence, exploring weakly the feature space is a natural extension of these algorithms that should conserve their advantages.
- With boosting, it is well known that the stronger the base learner is, the higher the chances of overfitting are. When embedding automatic feature generation into boosting, this problem is particularly relevant since the feature space may be highly expressive. Weak exploration of the feature space is a way to ensure that, even with very expressive candidate features, the base learner remains weak, hence reducing the risk of overfitting.

In the context of ensemble models, we therefore suggest that it may be unnecessary or even counter-productive to use computationally intensive optimization algorithms (e.g. genetic programming, or sophisticated heuristic search) as traditionally done in automatic feature generation. To explore this idea, the following section proposes fast randomized feature generation algorithms invoked locally during tree growing and in Section 5 we study them empirically.

4 Feature Generation Algorithms

We define first the feature grammar, then their generation as a sequential decision-making problem, and finally address this problem by Monte Carlo search.

4.1 Feature Grammar Using Reverse Polish Notation

Reverse polish notation (RPN) is a representation of expressions wherein every operator follows all of its operands. For instance, the RPN representation of the feature $c \times (a + b)$, where a, b and c are input properties and $+$ and $\times$ are constructor functions, is the sequence of symbols $[c, a, b, +, \times]$. This way of representing expressions is also known as postfix notation and is parenthesis-free

Algorithm 1. RPN evaluation

Require: $f \in \mathcal{A}^D$: a feature of size D
Require: $\mathbf{x} \in \mathcal{X}$: input properties
  stack $\leftarrow \emptyset$
  for $d = 1$ to D **do**
    if α_d is an input property **then**
      Push the value of α_d onto the stack
    else
      Let k be the arity of constructor α_d
      if $|stack| < k$ **then**
        syntax error
      else
        Pop the top k values from the stack,
        apply α_d to these operands and push the result onto the stack
      end if
    end if
  end for
  if $|stack| \neq 1$ **then**
    syntax error
  else
    return top(stack)
  end if

as long as operator arities are fixed, which makes it simpler to manipulate than its counterparts, prefix notation and infix notation.

Let $\mathcal{A}$ be the set of symbols composed of input properties and constructor functions. A feature f is a finite sequence of symbols of $\mathcal{A}$: $f = [\alpha_1, \ldots, \alpha_D] \in \mathcal{A}^*$. The size of a feature f is its number D of symbols. The evaluation of an RPN sequence relies on a stack and is depicted in Algorithm 1. This evaluation fails either if the stack does not contain enough operands when a constructor function is used or if the stack contains more than one single element at the end of the process. Feature $[a, \times]$ leads to the first kind of error: the function $\times$ of arity 2 is applied with a single operand. Feature $[a, a, a]$ leads to the second kind of error: evaluation finishes with three different elements on the stack. Features avoiding both kinds of errors are syntactically correct and are denoted $f \in \mathcal{F} \subset \mathcal{A}^*$.

4.2 Feature Generation as a Sequential Decision-Making Problem

We rely on RPN to formalize feature generation as a sequential decision-making problem. Thanks to this formalization, feature generation can be considered as a "one-player game" and solved using Monte Carlo search algorithms. In our approach, we expect the user to provide D, the maximal constructed feature size, i.e. the length of the longest features which will be considered as candidates. Given D, our sequential decision-making problem is defined as follows:

- **State space:** a state s is an RPN subsequence: $s = [\alpha_1, \ldots, \alpha_d] \in \mathcal{A}^*$ with $d \leq D$. The initial state is the empty sequence $s_0 = \emptyset$.

Table 2. Set of valid actions depending on the current state. Symbols are classified into *I*nput properties, *U*nary function constructors and *B*inary function constructors. $|stack|$ denotes the size of the current stack and d the length of the current RPN sequence. If the stack does not contain at least one element (resp. two elements), the unary functions (resp. binary functions) are excluded. When approaching the horizon D, input properties are excluded, or binary functions are forced to avoid finishing with too many elements on the stack.

State	Valid actions
$\|stack\| = 0$	I
$\|stack\| = 1$ & $d \neq D - 1$	I,U,$\perp$
$\|stack\| = 1$ & $d = D - 1$	U,$\perp$
$\|stack\| \in [2, D - d[$	I,U,B
$\|stack\| = D - d$	U,B
$\|stack\| = D - d + 1$	B

- **Action space:** the action space is $\mathcal{A} \cup \{\perp\}$, where $\perp$ is a special symbol to denote the end of a sequence. Given state s, we only consider a subset $\mathcal{A}_s^D \subset \mathcal{A}$ of valid actions to avoid the two syntax errors described earlier and to respect the constructor function typing constraints. Table 2 illustrates the set of valid actions $\mathcal{A}_s^D \subset \mathcal{A}$ in a simple case containing only unary and binary constructor functions that all operate on the same domain (e.g. only functions operating on real numbers). The following pre-processing can be used to compute the sets $\mathcal{A}_s$ in the general case. First, generate a tree of all the possible states of the stack for depths $d = 0, 1, \ldots, D$. The state of the stack is composed of a vector of variable domains; it does not depend on any particular input $\mathbf{x} \in \mathcal{X}$. Then, prune this tree by removing recursively all nodes leading to no valid RPN sequences. Given a particular state s, identify the corresponding state of the stack in this pruned tree and build $\mathcal{A}_s$ accordingly.
- **Transition function:** if the action $\perp$ is selected, we enter a final state defining feature $f = s$. Otherwise, the selected symbol is appended to the current RPN subsequence s.
- **Reward:** it is obtained when entering a final state and corresponds to the score computed by the target learning algorithm (see Table 1).

4.3 Monte Carlo Search for Feature Generation

Monte Carlo search algorithms for making optimal decisions are receiving an increasing interest in various fields of artificial intelligence [4], essentially due to their ability to combine the precision of tree search with the generality of random simulations. As a first step towards studying the application of these techniques to the problem of embedded feature generation, we focus here on three very simple Monte Carlo strategies: *random*, *step* and *look-ahead*.

These strategies are depicted in Algorithm 2. We denote by K the budget allowed to the search algorithm, i.e. the number of different features it can

Algorithm 2. Random, step and look-ahead feature generation algorithms

Require: budget K
Require: maximal feature size D

function RANDOMSIMULATION(state s)
  while s is not a final state **do**
    $a \sim \mathcal{U}_{\mathcal{A}_s^D}$ ▷ Sample valid action randomly
    $s \leftarrow s :: a$ ▷ Append symbol to s
  end while
  return s
end function

function STEPSIMULATION(state s)
  $r^* \leftarrow -\infty$; $f^* \leftarrow \emptyset$; $d \leftarrow 1$
  while s is not a final state **do**
    $f \leftarrow$ randomSimulation(s) ▷ Fill with random simulation
    if $score(f) > r^*$ **then** $f^* \leftarrow f$; $r^* \leftarrow score(f)$ **end if**
    $s \leftarrow s :: f^*[d]$ ▷ Append the d-th symbol of the best constructed feature
    $d \leftarrow d + 1$
  end while
  return f^*
end function

function LOOKAHEADSIMULATION(state s)
  $r^* \leftarrow -\infty$; $f^* \leftarrow \emptyset$; $d \leftarrow 1$
  while s is not a final state **do**
    for each $a \in \mathcal{A}_s^D$ **do**
      $f \leftarrow$ randomSimulation($s :: a$) ▷ Fill with random simulation
      if $score(f) > r^*$ **then** $f^* \leftarrow f$; $r^* \leftarrow score(f)$ **end if**
    end for
    $s \leftarrow s :: f^*[d]$ ▷ Append the d-th symbol of the best constructed feature
    $d \leftarrow d + 1$
  end while
  return f^*
end function

$r^* \leftarrow -\infty$; $f^* \leftarrow \emptyset$
while num evaluated features $< K$ **do**
  $f \leftarrow$ {random|step|lookAhead}Simulation($\emptyset$)
  if $score(f) > r^*$ **then** $f^* \leftarrow f$; $r^* \leftarrow score(f)$ **end if**
end while
return f^*

evaluate before to answer. The *random* strategy randomly generates K features, computes the score of each of these features and returns the best one. The two other algorithms start with an empty feature $s = \emptyset$ and proceed iteratively. At iteration d, the *step* strategy completes the current subsequence randomly and evaluates the corresponding feature. It then selects its next symbol as the d-th symbol of the currently best found feature f^*. Strategy *look-ahead* proceeds similarly, but makes more random simulations: one simulation per candidate

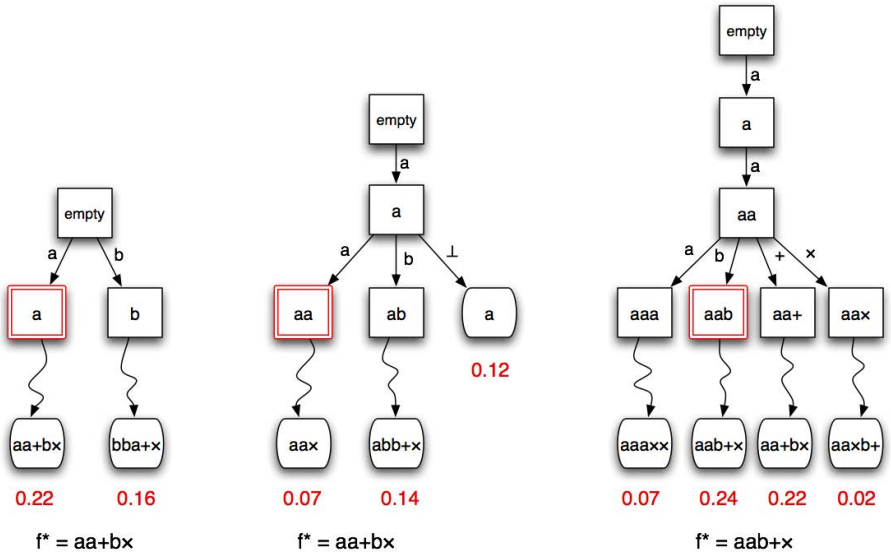


Fig. 2. Illustration of three steps of Algorithm 2 with the *look-ahead* strategy. Boxes denote states, curved edges denote random simulations and rounded boxes denote final states, i.e. constructed and evaluated features. Double-circled boxes denote the symbols of the best discovered feature so-far that are selected by the algorithm. Note that at the second step, the best discovered feature is still $[a, a, +, b, \times]$, hence the selected symbol is a.

successor symbol. This strategy corresponds to the level 1 nested Monte Carlo search algorithm [3]. Whatever the search strategy, our top-level algorithms work by repeatedly running the *random*, *step* or *look-ahead* strategy until K different features have been evaluated, i.e. search is stopped as soon as the split score function has been called K times. It then returns the best found feature.

Figure 2 illustrates three steps of our feature generation algorithm using the *look-ahead* strategy in a simple case with two input properties a and b and two constructor functions $+$ and $\times$.

5 Experimental Results

We validate our approach by embedding feature generation into five well-known classification algorithms: single trees, random forests, extremely randomized trees, boosted stumps and boosted trees. We focus on a set of 12 standard classification datasets, study the effect of the parameters D and K and compare algorithms embedding feature generation with their classical counter-parts.

5.1 Datasets and Methods

We use the same set of 12 multi-class classification datasets as the authors of [7]. These well-known and publicy available datasets cover a wide range of conditions

in terms of number of candidate attributes, presence of noise, non-linearity, observation redundancy and irrelevant variables. We also use the same train/test splits and the same evaluation protocol as [7]: for each dataset and each algorithm, we measure the test classification error averaged over 50 train/test splits for the smaller datasets and 10 train/test splits for the larger datasets.

We embed feature generation into the following algorithms. Single trees (ST) are classical decision trees without pruning. Random forests (RF) [1] and extremely randomized trees (ET) [7] are well-known ensemble methods. Boosted stumps (BS) and boosted decision trees (BT) rely on the AdaBoost.MH algorithm for multi-class classification [17]. For the ST, RF and ET methods, we use the information gain normalized in the same way as [7]. As in [9], when splitting nodes into boosted decision trees, we do not maximize information gain but rather directly maximize the *edge*, the objective of the weak learner in AdaBoost.MH.

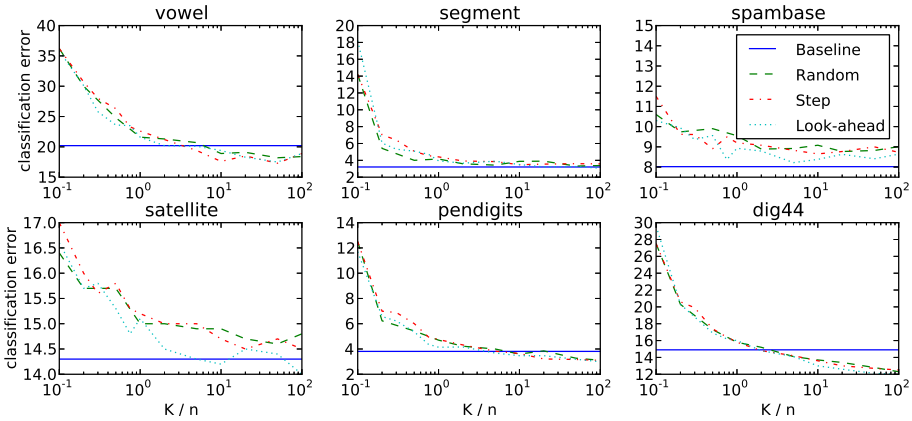
Although our formalism enables to deal with input properties and constructor functions of different types, we focus here on a simpler case: all input properties are numerical attributes and we construct features by using only the four mathematical operations $+, \times, -, /$. Applying our approach to more complex situations is left for future work.

5.2 Impact of Parameters K and D

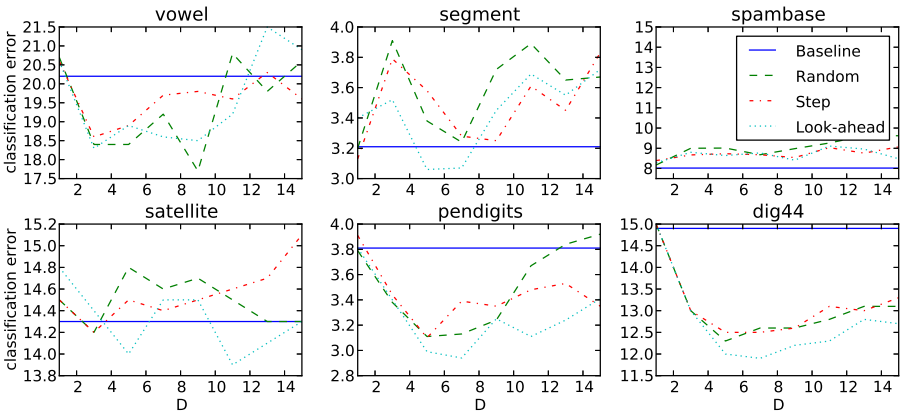
Our first series of experiments aims at studying the impact of the optimization budget K and the maximal feature size D on the performance of single trees, extremely randomized trees and boosted stumps. To study the impact of K, we choose a constant value $D = 5$, which for example enables to construct features such as $a + b \times c$ and vary K from $0.1n$ to $100n$. We then set K to a constant value (either $K = 100n$ for single trees or $K = 10n$ for the other methods) and vary the maximal feature length D from 1 to 15.

Single Trees. Figure 3 reports the results for single trees. The baseline corresponds to classical decision trees learned on the original variables of the problem. First, we observe that feature generation does not systematically lead to improved decision trees, which was already observed in previous work on feature generation. Second, we observe that scores continuously get better when the optimization budget K is increased. This is in agreement with the traditional approach to feature generation involving computationally intensive search algorithms such as genetic programming. Since there is a single chance to build a good-performing tree (there is no ensemble effect), as much computational power as possible should be dedicated to feature search. There is no clear tendency concerning the parameter D, although we see that $D = 5$ seems to be a reasonable default value. The *look-ahead* strategy slightly outperforms the two other search strategies, although the overall difference is rather small.

Extremely Randomized Trees. Figure 4 displays the results for ensembles of 100 extremely randomized trees. Our approaches are compared against traditional



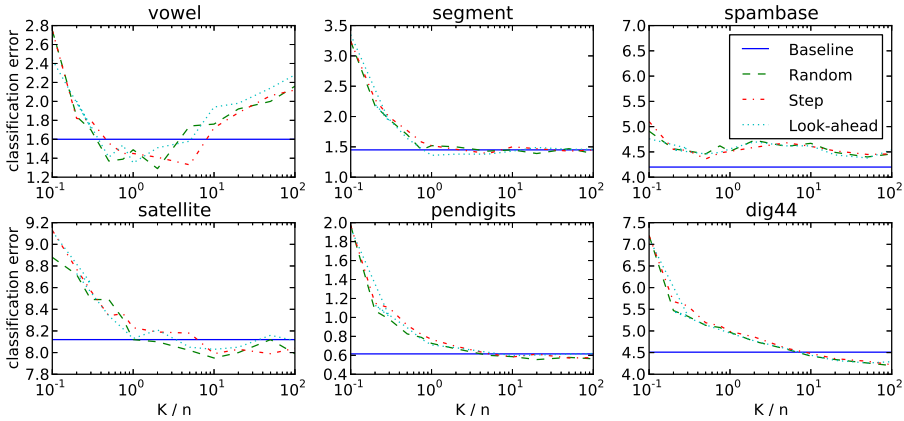
(a) Impact of parameter K with $D = 5$



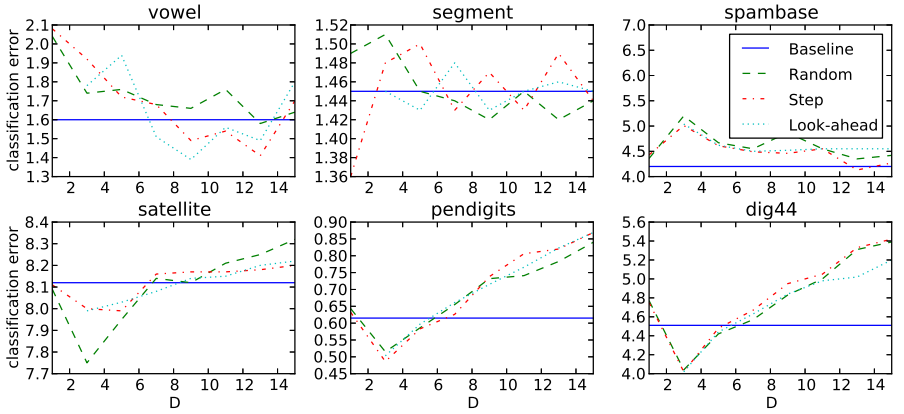
(b) Impact of parameter D with $K = 100n$

Fig. 3. Single trees

extremely randomized trees with parameter K tuned to give the best test scores. As previously, feature generation enables to obtain significantly improved models on only a part of the datasets. On the four first datasets, the best scores are obtained for values of K ranging from $0.1n$ to $10n$. As discussed in Section 3, extremely randomized trees (and random forests) rely on random subspaces to introduce randomization, which has been shown to lead to more robust ensemble models. We observe that, when extending these algorithms with automatic feature generation, it is still interesting to use random subspaces, i.e. to explore a very small portion of the whole feature space at each node split. This phenomenon is particularly clear on the vowel dataset, for which we observe that investing too much computational power into feature search is counterproductive. On the three last datasets, our approach seems to work the best for small generated features of size $D = 3$. Again, there is very little difference between the three search strategies.



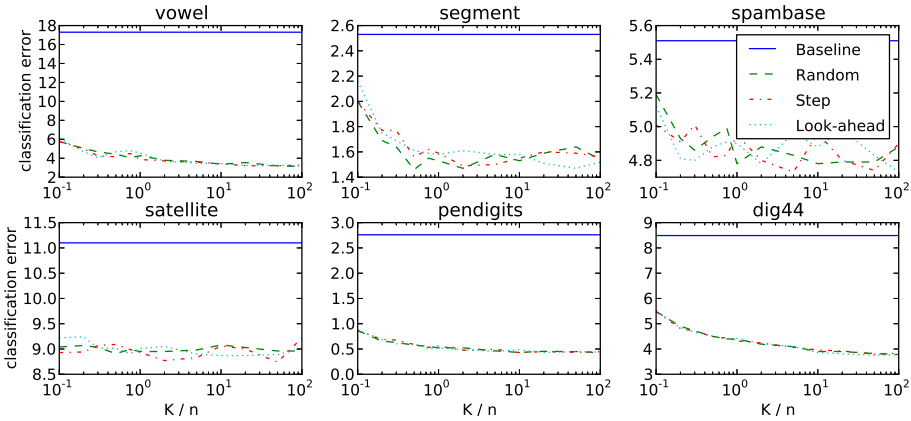
(a) Impact of parameter K with $D = 5$



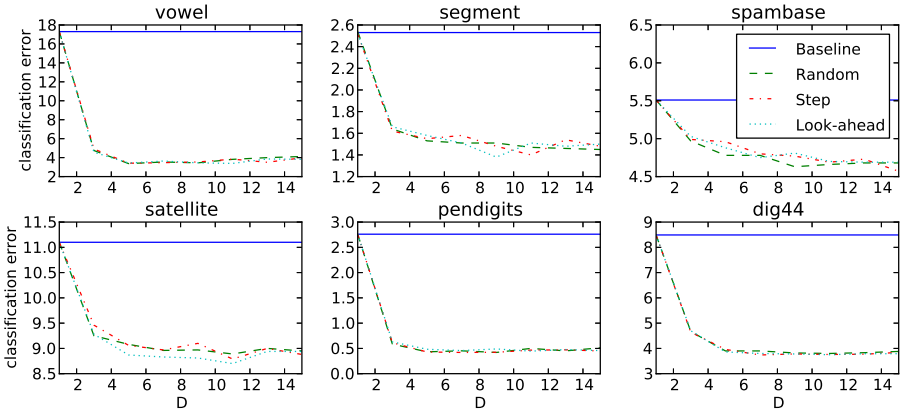
(b) Impact of parameter D with $K = 10n$

Fig. 4. Extremely randomized trees

Boosted Stumps. Figure 5 presents the results for ensembles of 1000 boosted stumps. We now observe that models with feature generation strongly outperform classical boosted stumps on all datasets. We believe that this is mainly explained by the fact that the baseline model is not able to exploit multi-variate correlations, whereas our extended version can exploit multiple variables together thanks to generated features. The most impressive in these results is that, even with very small computational budgets $K = 0.1n$ that correspond to testing one or a few features at each iteration, feature generation still yields significantly improved models. Furthermore, we observe that increasing K beyond $10n$ only brings slight or no improvements on the error. In these cases, there is thus no need to invest a huge amount of computational power in feature search to benefit from feature generation. We also observe that the method is here rather robust w.r.t. the choice of parameter D and again that differences between the three search strategies are small.



(a) Impact of parameter K with $D = 5$



(b) Impact of parameter D with $K = 10n$

Fig. 5. Boosted stumps

5.3 Overall Comparison of Methods

The results of embedding feature generation with our three search strategies into our five supervised learning algorithms are given in Table 3 for all our 12 datasets. For all methods, we use default settings for hyper-parameters: the maximal feature size is set to $D = 5$ in all cases, the feature search budget is set to $K = 100n$ for single trees and $K = 10n$ for ensemble models. The baseline random forests are constructed with $K^{RF} = \sqrt{n}$ tested attributes per constructed splits. For extremely randomized trees, we consider the same two default setups as in [7]: $K^{ET} = \sqrt{n}$ and $K^{ET} = n$.

We observe that embedding the Monte Carlo feature generation improves over the baselines about two times out of three, and on the average all methods are significantly improved thanks to the feature generation. Among the variants we

Table 3. Comparison of all methods with and without embedded feature generation. The scores (error rates in %) of methods embedding feature generation are shown in bold whenever they outperform those of the corresponding baseline(s). The best scores for each dataset are underlined.

	Wave form	Two norm	Ring norm	Vehicle	Vowel	Segment	Spam base	Satellite	Pen digits	Dig44	Letter	Isolet	mean
	Single Tree, $K = 100n$, $D = 5$												
Baseline	29.2	21.6	16.6	20.9	20.2	3.21	8.02	14.3	3.81	14.9	14.9	25.8	16.12
Random	**25.9**	**15.6**	17.5	21.9	**18.4**	3.38	9.01	14.8	**3.11**	**12.3**	15.2	**25.6**	**15.23**
Step	**26.0**	**15.3**	17.3	21.9	**18.9**	3.58	8.73	14.5	**3.10**	**12.5**	15.6	**24.7**	**15.17**
Look-ahead	**25.3**	**15.3**	18.1	21.7	**18.9**	**3.06**	8.64	**14.0**	**2.99**	**12.0**	14.9	**22.0**	**14.74**
	Random Forests, 1000 trees, $K = 10n$, $D = 5$												
$K^{RF} = \sqrt{n}$	17.1	4.08	6.03	23.5	3.27	1.94	4.57	8.46	0.969	5.27	4.62	7.78	7.30
Random	**15.6**	**2.89**	**2.92**	**20.3**	3.43	**1.72**	4.71	8.46	**0.606**	**4.12**	**3.74**	7.56	**6.34**
Step	**15.6**	**2.87**	**2.78**	**20.1**	3.35	**1.71**	4.62	**8.45**	**0.606**	**4.05**	**3.75**	7.63	**6.29**
Look-ahead	**15.7**	**2.92**	**2.93**	**20.1**	3.54	**1.69**	4.68	8.46	**0.615**	**4.02**	**3.81**	**7.46**	**6.33**
	Extra Trees, 1000 trees, $K = 10n$, $D = 5$												
$K^{ET} = \sqrt{n}$	16.1	3.08	2.88	24.0	1.51	1.85	4.31	8.33	0.652	4.25	3.53	6.75	6.43
$K^{ET} = n$	17.4	4.74	5.23	22.0	1.92	1.42	4.31	7.99	0.626	4.59	4.01	7.85	6.85
Random	**15.8**	**2.77**	**2.44**	**20.4**	1.58	**1.41**	4.56	**7.85**	**0.555**	4.29	**3.23**	6.98	**5.99**
Step	**15.7**	**2.72**	**2.29**	**20.4**	1.58	1.43	4.57	**7.85**	**0.578**	4.36	**3.24**	7.00	**5.98**
Look-ahead	**15.8**	**2.78**	**2.38**	**20.5**	1.62	1.45	4.44	**7.89**	**0.589**	4.31	**3.21**	6.96	**5.99**
	Stump Boosting, 1000 iterations, $K = 10n$, $D = 5$												
Baseline	19.6	4.88	9.94	20.0	17.3	2.53	5.51	11.1	2.76	8.49	17.9	8.52	10.72
Random	**17.0**	**3.18**	**5.08**	**18.3**	**3.37**	**1.53**	**4.78**	**9.08**	**0.432**	**3.90**	**6.36**	**6.40**	**6.62**
Step	**17.1**	**3.16**	**5.07**	**18.1**	**3.41**	**1.55**	**4.96**	**9.06**	**0.429**	**3.96**	**6.43**	**6.50**	**6.64**
Look-ahead	**17.0**	**3.14**	**5.09**	**17.9**	**3.43**	**1.58**	**4.88**	**8.87**	**0.483**	**3.86**	**6.27**	**6.24**	**6.56**
	Tree Boosting, 1000 iterations, Depth 3, $K = 10n$, $D = 5$												
Baseline	17.1	3.72	8.84	19.7	2.44	1.09	4.97	8.09	0.472	3.56	4.34	5.72	6.67
Random	**15.6**	**2.89**	**3.67**	**18.2**	**2.00**	1.32	**4.66**	**7.92**	**0.375**	**2.96**	**3.38**	**5.51**	**5.71**
Step	**15.6**	**2.89**	**3.61**	**18.4**	**2.08**	1.32	**4.62**	**7.74**	**0.349**	**2.97**	**3.30**	**5.51**	**5.71**
Look-ahead	**15.6**	**2.90**	**3.65**	**18.6**	**2.18**	1.31	**4.52**	**7.77**	**0.397**	**2.87**	**3.28**	**5.40**	**5.71**
	Tree Boosting, 1000 iterations, Depth 5, $K = 10n$, $D = 5$												
Baseline	16.6	3.66	6.82	21.2	1.90	1.23	4.85	7.55	0.460	3.27	3.53	5.91	6.41
Random	**15.4**	**2.84**	**3.10**	**20.0**	1.98	1.36	**4.60**	7.74	**0.403**	**2.99**	**3.00**	6.21	**5.80**
Step	**15.4**	**2.82**	**3.05**	**19.7**	2.00	1.42	**4.52**	7.76	**0.397**	**3.00**	**3.00**	6.26	**5.78**
Look-ahead	**15.4**	**2.85**	**3.05**	**19.7**	1.90	1.39	**4.53**	7.90	**0.392**	**2.99**	**2.95**	6.14	**5.76**

tested, the overall best scores are obtained when embedding feature generation in boosted decision trees of depth 3. The strongest improvement is however observed for boosted stumps, where we believe that there is a combined bias and variance reduction effect obtained thanks to the randomized feature construction. The net result is that, on the average, combining stump boosting with feature generation becomes competitive with baseline boosted trees of depth 3 or 5. Finally, while the ensemble methods are well improved when using a rather small budget for the search of features (K= $10n$), standard trees may be improved by using a quite larger search budget (K= $100n$).

6 Conclusion

Automatic feature generation approaches are usually classified into filters and wrappers. This paper emphasizes a third category: *embedded* feature generation. We have proposed a general scheme to embed feature generation into tree-based learning algorithms and have discussed the particularities of feature generation

in the context of ensemble methods. In this latter context, we argued that it could be unnecessary or even counter-productive to invest a too large amount of computational power into feature search and therefore proposed three simple Monte Carlo search approaches to at the same time weakly and efficiently explore the feature space, the number of trials allowing to control search strength. Our empirical investigations confirmed this analysis and showed also that the embedding of feature generation allows to improve the accuracy over a wide range of methods and datasets, the strongest improvements being found in the context of boosting, where Monte Carlo feature generation allows to both reduce bias (in the context of stumps) and variance (in the context of all versions).

For future research, we suggest to revisit in the light of our findings the earlier work on oblique decision trees (see e.g. [12]), which may also be viewed as a kind of embedded feature generation. On the other hand, while we focused in this paper on accuracy improvement, we believe that embedded feature generation should also be studied in terms of its effects on model complexity (e.g. in terms of the speed of convergence of the ensemble methods) and on interpretability (e.g. in terms of the capability to detect variable interactions of interest). Another line of research would be to see how to port the embedded feature generation towards L1-based regularization of generalized additive models, by building on the parallels of these approaches with boosting.

The formalism presented in this paper is very general in that it enables to formulate embedded feature generation on top of all kinds of raw data structures, including functional signals such as audio and images, and graph structured or relational data more and more frequently found in current application domains (e.g. ranging from bio-informatics to web-mining or automatic game playing). While our experiments focused on constructing features based on raw vectors of simple numerical features, we believe that embedded feature generation has even greater potentials to learn with such complex real-world data structures.

Acknowledgements. This paper presents research results of the European excellence network PASCAL2 and of the Belgian Network BIOMAGNET, funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with the authors.

References

1. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
2. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: ICML 2006, pp. 161–168. ACM, New York (2006)
3. Cazenave, T.: Nested monte-carlo search. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 456–461 (2009)
4. Chaslot, G., Bakkes, S., Szita, I., Spronck, P.: Monte-carlo tree search: A new framework for game ai. In: Darken, C., Mateas, M. (eds.) Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (2008)
5. Ekárt, A., Márkus, A.: Using genetic programming and decision trees for generating structural descriptions of four bar mechanisms. Artif. Intell. Eng. Des. Anal. Manuf. 17(3), 205–220 (2003)

6. Espejo, P.G., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. Trans. Sys. Man Cyber. Part C 40(2), 121–144 (2010)
7. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning 36(1), 3–42 (2006)
8. Guo, H., Jack, L.B., Nandi, A.K.: Feature generation using genetic programming with application to fault classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B 35(1), 89–99 (2005)
9. Kégl, B., Busa-Fekete, R.: Boosting products of base classifiers. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, pp. 497–504. ACM, New York (2009)
10. Krzysztof, K.: Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genetic Programming and Evolvable Machines 3(4), 329–343 (2002)
11. Markovitch, S., Rosenstein, D.: Feature generation using general constructor functions. Machine Learning 49, 59–98 (2002)
12. Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trrees. Journal of Artificial Intelligence Research 2, 1–32 (1994)
13. Ng, A.Y.: Feature selection, L1 vs. L2 regularization, and rotational invariance. In: Proceedings of the Twenty-First International Conference on Machine Learning, ICML 2004. ACM, New York (2004)
14. Pachet, F., Roy, P.: Analytical features: a knowledge-based approach to audio feature generation. EURASIP J. Audio Speech Music Process., 1–23 (2009)
15. Pagallo, G., Haussler, D.: Boolean feature discovery in empirical learning. Machine Learning 5(1), 71–99 (1990)
16. Raymer, M.L., Punch, W.F., Goodman, E.D., Kuhn, L.A.: Genetic programming for improved data mining: application to the biochemistry of protein interactions. In: Proceedings of the First Annual Conference on Genetic Programming, GECCO 1996, pp. 375–380. MIT Press, Cambridge (1996)
17. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37(3), 297–336 (1999)
18. Smith, M.G., Bull, L.: Genetic programming with a genetic algorithm for feature construction and selection. Genetic Programming and Evolvable Machines 6(3), 265–281 (2005)
19. Yang, D.-S., Rendell, L., Blix, G.: A scheme for feature construction and a comparison of empirical methods. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, pp. 699–704. Morgan Kaufmann (1991)

Hypergraph Spectra for Semi-supervised Feature Selection

Zhihong Zhang[1], Edwin R. Hancock[1,⋆], and Xiao Bai[2]

[1] Department of Computer Science,
University of York, UK
[2] School of Computer Science and Engineering
Beihang University, China

Abstract. In many data analysis tasks, one is often confronted with the problem of selecting features from very high dimensional data. Most existing feature selection methods focus on ranking individual features based on a utility criterion, and select the optimal feature set in a greedy manner. However, the feature combinations found in this way do not give optimal classification performance, since they neglect the correlations among features. While the labeled data required by supervised feature selection can be scarce, there is usually no shortage of unlabeled data. In this paper, we propose a novel hypergraph based semi-supervised feature selection algorithm to select relevant features using both labeled and unlabeled data. There are two main contributions in this paper. The first is that by incorporating multidimensional interaction information (MII) for higher order similarities measure, we establish a novel hypergraph framework which is used for characterizing the multiple relationships within a set of samples. Thus, the structural information latent in the data can be more effectively modeled. Secondly, we derive a hypergraph subspace learning view of feature selection which casting the feature discriminant analysis into a regression framework that considers the correlations among features. As a result, we can evaluate joint feature combinations, rather than being confined to consider them individually. Experimental results demonstrate the effectiveness of our feature selection method on a number of standard face data-sets.

Keywords: Hypergraph representation, Semi-supervised subspace learning.

1 Introduction

In order to render the analysis of high-dimensional data tractable, it is crucial to identify a smaller subset of features that are informative for classification and clustering. Dimensionality reduction aims to reduce the number of variables under consideration, and the process can be divided into feature extraction and feature selection. Feature extraction usually projects the features onto a low-dimensional and distinct feature space, e.g., Locally Linear Embedding (LLE)

⋆ Edwin Hancock is supported by a Royal Society Wolfson Research Merit Award.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 207–222, 2012.

[1], kernel PCA [2], Locality preserving Projection (LPP) [3], Neighborhood Preserving Embedding (NPE) [4] and Laplacian eigenmap [5]. Unlike feature extraction, feature selection identifies the optimal feature subset in the original feature space. By maintaining the original features, feature selection improves the interpretability of the data, which is preferred in many real world applications, such as face recognition [28]. Feature selection algorithms can be roughly classified into three groups, namely a) supervised feature selection, b) unsupervised feature selection and c) semi-supervised feature selection.

Supervised feature selection algorithms usually evaluate the importance of features using the correlation between the features and the class label. An important line of research in this area is the use of methods based on mutual information. For example, Battiti [6] has developed the Mutual Information-Based Feature Selection (MIFS) criterion, where the features are selected in a greedy manner. Given a set of existing selected features S, at each step it locates the feature x_i that maximizes the relevance to the class i.e. $I(x_i; C)$. The selection is regulated by a proportional term $\beta I(x_i; S)$ that measures the overlap information between the candidate feature and the existing features. The parameter β may significantly affect the features selected, and its control remains an open problem. Peng et al. [7] on the other hand, use the so-called Maximum-Relevance Minimum-Redundancy criterion (MRMR), which is equivalent to MIFS with $\beta = \frac{1}{n-1}$.

Supervised spectral feature selection algorithms provide powerful alternatives to those discussed above. Examples include the Fisher score [8], the trace ratio [9] and ReliefF [10]. Given d features, and a similarity matrix S for the samples, the idea underpinning spectral feature selection algorithms is to identify features that align well with the leading eigenvectors of S. The leading eigenvectors of S contain information of concerning the structure of the sample distribution and group similar samples into compact clusters. Consequently, features that align closely to the eigenvctors will better preserve sample similarity [11].

While the labeled data required by supervised feature selection can be scarce, there is usually no shortage of unlabeled data. Hence, there are obvious attractions in developing unsupervised feature selection algorithms which can utilize this data. The typical examples in unsupervised learning are graph-based spectral learning algorithms. Examples include the Laplacian score [12], SPEC [11], Multi-Cluster Feature Selection (MCFS) [22] and Unsupervised Discriminative Feature Selection (UDFS) [23]. A frequently used criterion in unsupervised graph-based spectral learning is to select the features which best preserve the structure of the data similarity or a manifold structure derived from the entire feature set. For example, the Laplacian score [12] uses a nearest neighbor graph to model the local geometric structure of the data, using the pairwise similarities between features calculated using the heat kernel. In this framework, the features are evaluated individually and are selected one by one. Another unsupervised spectral feature selection algorithm is SPEC [11], which is an extension of the Laplacian score aimed at making it more robust to noise. The method selects

the features that are most consistent with the graph structure. Note that SPEC also evaluates features independently.

However, there are some drawbacks with the above graph-based spectral learning methods when they are used to deal with computer vision problems. One of these is that they evaluate features individually and hence cannot handle redundant features. Redundant features increase the dimensionality unnecessarily, and degrade learning performance when faced with a shortage of data. It is also shown empirically that removing redundant features can result in significant performance improvement. Another problem is that their graph representations are very sensitive to the topological structure and lack sufficient robustness in real-world learning tasks. This is because in real world problems the objects and their features tend to exhibit multiple relationships rather than simple pairwise ones. This factor will considerably compromise the performance of learning approaches. For example, consider the problem of grouping images of five different persons based on identity, each of which is imaged in the same pose, but under different lighting conditions. See Fig. 1 for an illustration. Recent studies on illumination have shown that images of the same objects may look drastically different under different lighting conditions while different objects may appear similar under different illumination conditions [24]. Furthermore, it is well known that the set of images of a Lambertian surface under arbitrary lighting (without shadowing) lies on a 3-D liner subspace in the image space [13]. As any three images span a 3-D subspace, one needs to consider at least four images at a time to define a measure of similarity. Therefore, suitable for graph construction and similarity measure are needed.

Fig. 1. Shown above are images of five persons under varying illumination conditions

A natural way of remedying the information loss described above is to represent the data set as a hypergraph instead of a graph. Hypergraph representations allow vertices to be multiply connected by hyperedges. They can hence be both accurate and complete in describing feature relations and structures. Due to their effectiveness in representing multiple relationships, in this paper, we propose a hypergraph based semi-supervised feature selection method which can be used to classify face images under varying illumination conditions. This method jointly evaluates the utility for sets of features rather than individual features. There are three novel ingredients. The first is that by incorporating hypergraph representation into feature selection, we can be more effective capture the higher order relations among samples. Secondly, inspired from the recent works on mutual

information [26], [27], we determine the weight of a hyperedge using an information measure referred to as multidimensional interaction information (MII) which precisely preserves the higher order relations captured by the hypergraph. The advantage of MII is that it is sensitive to the relations between sample combinations, and as a result can be used to seek third or even higher order dependencies among the relevant samples. Thus, the structural information latent in the data can be more effectively modeled. Finally, we describe a new semi-supervised feature selection strategy through hypergraph subspace learning, which casts the feature discriminant analysis into a regression framework that considers the correlations among features. As a result, we can evaluate joint feature combinations, rather than being confined to consider them individually, thus it is able to handle feature redundancy.

2 Hypergraph Construction

In this section, we establish a novel hypergraph framework which is used for characterizing the multiple relationships within a set of samples. To this end, we commence by introducing a new method for measuring higher order similarities among samples based on information theory. According to Shannon's study, the uncertainty of a random variable X can be measured by the entropy $H(X)$. For two random variables X and Y, the conditional entropy $H(Y|X)$ measures the remaining uncertainty about Y when X is known. The mutual information $I(X;Y)$ of X and Y quantifies the information gain about Y provided by X. The relationship between $H(Y)$, $H(Y|X)$ and $I(X;Y)$ is $I(X;Y) = H(Y) - H(Y|X)$. As defined by Shannon, the initial uncertainty for X is $H(X) = -\sum_{x\in Y} P(x)\log P(x)$, where $P(x)$ is the prior probability density function over $x \in X$. The remaining uncertainty for Y if X is known is defined by the conditional entropy $H(Y|X) = -\int_x p(x)\{\sum_{y\in Y} p(y|x)\log p(y|x)\}dx$, where $p(y|x)$ denotes the posterior probability for $y \in Y$ given $x \in X$. After observing x, the amount of additional information gain is given by the mutual information

$$I(X;Y) = \sum_{y\in Y}\int_x p(y,x)log\frac{p(y,x)}{p(y)p(x)}dx \ . \tag{1}$$

The mutual information (1) quantifies the information which is shared by X and Y. When the $I(X;Y)$ is large, it implies that x and y are closely related. Otherwise, when $I(X;Y)$ is equal to 0, it means that two variables are totally unrelated. Analogically, the conditional mutual information of X and Y given Z, denoted as $I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$, represents the quantity of information shared by X and Y when Z is known. The conditioning on a third random variable may either increase or decrease the original mutual information. In this context, the Interaction Information $I(X;Y;Z)$ is defined as the difference between the conditional mutual information and the simple mutual information, i.e.

$$I(X;Y;Z) = I(X;Y|Z) - I(X;Y) \ . \tag{2}$$

The interaction information $I(X;Y;Z)$ measures the influence of the variable Z on the amount of information shared between variables X and Y. Its value can be positive, negative, or zero. Zero valued Interaction Information $I(X;Y;Z)$ implies that the relation between X and Y entirely depends on Z. A positive value of $I(X;Y;Z)$ implies that X and Y are independent of each other themselves, but are correlated with each other when combined with Z. A negative value of $I(X;Y;Z)$ indicates that Z can account for or explain the correlation between X and Y. The generalization of Interaction Information to K variables is defined recursively as follow

$$I(\{X_1, \cdots, X_K\}) = I(\{X_2, \cdots, X_K\}|X_1) - I(\{X_2, \cdots, X_K\}) \ . \tag{3}$$

Based on the higher order similarity measure, we establish a hypergraph framework for characterizing a set of high dimensional samples. A hypergraph is defined as a triplet $H = (V, E, w)$. Here V denotes the vertex set, E denotes the hyperedge set in which each hyperedge $e \in E$ represents a subset of V, and w is a weight function which assigns a real value $w(e)$ to each hyperedge $e \in E$. We only consider K-uniform hypergraphs (i.e. those for which the hyperedges have identical cardinality K) in our work. Given a set of high dimensional samples $\mathbf{X} = [x_1, \cdots x_N]^T$ where $x_i \in \mathbb{R}^d$, we establish a K-uniform hypergraph, with each hypergraph vertex representing an individual sample and each hyperedge representing the Kth order relations among a K-tuple of participating samples. A K-uniform hypergraph can be represented in terms of Kth order matrix, i.e. a tensor $\mathcal{W}$ of order K, whose element $W_{i_1,\cdots,i_K}$ is the hyperedge weight associated with the K-tuple of participating vertices $\{v_{i_1}, \cdots, v_{i_K}\}$. In our work, the hyperedge weight associating with $\{x_{i_1}, x_{i_2}, \cdots, x_{i_K}\}$ is computed as follows

$$W_{i_1,\cdots,i_K} = K \frac{I(x_{i_1}, x_{i_2}, \cdots, x_{i_K})}{H(x_{i_1}) + H(x_{i_2}) + \cdots H(x_{i_K})} \ . \tag{4}$$

It is clear that $W_{i_1,\cdots,i_K}$ is a normalized version of Kth order Interaction Information. The greater the value of $W_{i_1,\cdots,i_K}$ is, the more relevant the K samples are. On the other hand, if $W_{i_1,\cdots,i_K} = 0$, the K samples are totally unrelated.

3 Hypergraph Representation

Unlike matrix eigen-decomposition, there has not yet been a widely accepted method for spanning a rationale eigen-space for a tensor [30]. Therefore, it is hard to directly embed a hypergraph into a feature space spanned by its tensor representation through eigen-decomposition. In our work, we consider the transformation of a K-uniform hypergraph into a graph. Accordingly, the associated hypergraph tensor $\mathcal{W}$ is transformed to a graph adjacency matrix $\boldsymbol{A}$, and the higher order information exhibited in the original hypergraph can be encoded in an embedding space spanned by the related matrix representation. In this scenario, one straightforward way for the transformation is marginalization which

computes the arithmetical average over all the hyperedge weights $W_{i_1,\cdots,i_{K-2},i,j}$ associated with the edge weight $A_{i,j}$

$$\tilde{A}_{i,j} = \sum_{i_1=1}^{|V|} \cdots \sum_{i_{K-2}=1}^{|V|} W_{i_1,\cdots,i_{K-2},i,j} \tag{5}$$

The edge weight $\tilde{A}_{i,j}$ for edge ij is generated by a uniformly weighted sum of hyperedge weights $W_{i_1,\cdots,i_{K-2},i,j}$. However,the form appearing in (5) behaves as a low pass filter, and thus results in information loss through marginalization.

To make the process of marginalization more comprehensive, we use marginalization to constrain the sum of edge weights and then estimate their values through solving an over-constrained system of linear equations. Our idea is motivated by the so called *clique average* introduced in the higher order clustering literature [15]. We characterize the relationships between $\boldsymbol{A}$ and $\mathcal{W}$ as follows

$$W_{i_1,\cdots,i_K} = \sum_{\{i,j\}\subseteq\{i_1,\cdots,i_K\}} A_{i,j} \tag{6}$$

There are $\binom{|V|}{2}$ variables and $\binom{|V|}{K}$ equations in the system of equations described in (5). When $K > 2$, the linear system (5) is over-determined and cannot be solved analytically. We thus approximate the solution to (5) by minimizing the least squares error

$$\widehat{\boldsymbol{A}} = \underset{\boldsymbol{A}}{\operatorname{argmax}} \sum_{i_1,\cdots,i_K} \left(\sum_{\{i,j\}\subseteq\{i_1,\cdots,i_K\}} A_{i,j} - W_{i_1,\cdots,i_K} \right)^2 \tag{7}$$

In practical computation, we normalize the compatibility tensor $\mathcal{W}$ by using the extended Sinkhorn normalization scheme [31], and constrain the element of $\boldsymbol{A}$ to be in the interval $[0, 1]$ to avoid unexpected infinities. Effective iterative numerical methods are used to compute the approximated solutions [32].

The adjacency matrix $\boldsymbol{A}$ computed through (7) is one effective representation for a K-uniform hypergraph, because it naturally avoids the operation of arithmetic average and thus to a certain degree overcomes the low pass information loss arising in (5). Furthermore, the $\boldsymbol{D}$ is the diagonal matrix with its ith diagonal element being $A_{ii} = \sum_j A_{ij}$. In this context, a hypergraph can be easily embedded into a feature space spanned by a semi-supervised subspace learning, which will be explained in detail in the next Section.

4 Feature Selection through Semi-supervised Subspace Learning

In this section, we formulate the procedure of semi-supervised feature selection on a basis of hypergraph subspace learning. Feature selection can be seen as a special subspace learning task, where the projection matrix is constrained to be selection matrix.

4.1 Hypergraph Semi-supervised Subspace Learning

One goal of hypergraph subspace learning is to represent the high dimensional data $\mathbf{X} \in \mathbb{R}^{N\times d}$ by a low dimensional representation $\mathbf{Y} \in \mathbb{R}^{N\times C}$ ($C \ll d$) in the low dimensional feature space such that the structural characteristics of the high dimensional data are well preserved or are more "obvious". Here we use the representations $\mathbf{X} = [x_1, \cdots x_N]^T$ and $\mathbf{Y} = [y_1, \cdots, y_k, \cdots, y_C]$, where y_k is a N-dimensional vector and its N elements represent the N samples $x_1, \cdots x_N$ separately in the kth dimension of the low dimensional feature space. Based on the hypergraph transformation described in Section 3, the semi-supervised hypergraph subspace learning can be easily conducted as the following two steps, a) label propagation, b) label regression:

Label Propagation: First, we obtain the soft labels of unlabeled data using a new label propagation method [25], in which an additional class labeled $C+1$ is introduced to accommodate the outliers data. Give a data set $\mathbf{X} = \{x_1, \cdots, x_l, \cdots, x_N\} \subset \mathbb{R}^d$ and the first l data are labeled and the subsequent $u = N - l$ data are unlabeled. Define the initial label matrix $L = [(l_1)^T, (l_2)^T, \cdots, (l_N)^T] \in \mathbb{R}^{N\times(C+1)}$, for the labeled data, $L_{ij} = 1$ if x_i is labeled as $l_i = j$ and $L_{ij} = 0$ otherwise. For the unlabeled data, $L_{ij} = 1$ if $j = C+1$ and $L_{ij} = 0$ otherwise. Denote a stochastic matrix $\mathbf{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$, to assign labels to the unlabeled data, an iteration function $\mathbf{F}(t+1) = \alpha\mathbf{P}\mathbf{F}(t) + (1-\alpha)L$ is computed on the nodes of graph until convergence so as to satisfy two constraints, where α is a parameter in $(0, 1)$. During each iteration, each point receives information from its neighbors (first term), and also retains its initial information (second term). $\mathbf{F} = [\mathbf{F}_1^T, \ldots, \mathbf{F}_N^T]^T \in \mathbb{R}^{N\times(C+1)}$ is the predicted soft label matrix, where $\mathbf{F}_{ij}$ reflects the posterior probability of data point x_i belonging to class j [29]. When $j = C+1$, $\mathbf{F}_{i,C+1}$ represents the probability of x_i belonging to outlier. The iteration process will converge to the fixed point

$$\mathbf{F} = \lim_{t\to\infty} \mathbf{F}(t) = (1-\alpha)(\mathbf{I} - \alpha\mathbf{P})^{-1}L \ . \tag{8}$$

Using this iterative label propagation scheme, the outlier in data can be detected and the label for each unlabeled point is assigned to the class from which it has received the most information during the iteration process.

Label Regression: After we obtain the soft label $\mathbf{F}_i$ for each data x_i, i.e., the probability $\mathbf{F}_{ij}$ of x_i belonging to class j, we can learn the subspace for data using the soft labels. Assuming the low-dimensional data can be obtained from the linear projection $\mathbf{Y} = \mathbf{X}\mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{d\times C}$ is the projection vector, a regression function can be defined as

$$\arg\min_{\mathbf{W},b} \gamma\|\mathbf{W}\|^2 + \sum_{i=1}^{N}\sum_{j=1}^{C} \mathbf{F}_{ij}\|\mathbf{W}^T x_i + b - t_j\|^2 \ , \tag{9}$$

where the bias term $b \in \mathbb{R}^{C\times 1}$, $t_j = [0, \ldots, 0, 1, 0, \cdots, 0]^T$ is the class indicator vector for the j-th class and γ is the regularization parameter. It can be easily

verified that for larger values of $\mathbf{F}_{i,C+1}$, which indicates x_i is belonging to the outlier, then the values of $\mathbf{F}_{i,j}$ $(1 \leq i \leq C)$ should be smaller to reduce the effect of the outlier data point x_i in the regression model according to Equation (9). Differentiating the matrix form in Equation (9) w.r.t $\mathbf{W}$ and b, we have the solution to the regression problem is

$$\mathbf{W} = (\mathcal{X}\mathbf{L}_s\mathcal{X}^T + \gamma\mathbf{I})^{-1}\mathcal{X}\mathbf{C}_s\mathbf{F}_C \ , \tag{10}$$

where $\mathbf{I}$ denotes an identity matrix with proper size, $\mathbf{L}_s = \mathbf{S} - \frac{1}{1_N^T\mathbf{S}1_N}\mathbf{S}1_N1_N^T\mathbf{S}$, $\mathbf{S}_{ii} = \sum_{j=1}^{C}\mathbf{F}_{ij}$, $\mathbf{F}_C \in \mathbb{R}^{N\times C}$ is formed by the first C columns of $\mathbf{F}$, $\mathbf{C}_s = \mathbf{I} - \frac{1}{1_N^T\mathbf{S}1_N}\mathbf{S}1_N1_N^T$ and $1_N \in \mathbb{R}^{N\times 1}$.

4.2 Robust Feature Selection Based on ℓ_1-Norms

The hypergraph subspace learning procedure can be viewed as feature extraction, and can be expressed as $\mathbf{Y} = \mathbf{X}\mathbf{W}$ where $\mathbf{W} \in \mathbb{R}^{d\times C}$ is a column-full-rank projection matrix. However, unlike feature extraction, feature selection attempts to select the optimal feature subset in the original feature space. Therefore, for the task of feature selection, the projection matrix $\mathbf{W} = [w_1, \ldots w_C]$ can be constrained to be a selection matrix $\Phi = [\Phi_1, \ldots \Phi_C]$ which contains the combination coefficients for different features in approximating $\mathbf{Y} = [y_1, \ldots, y_C]$. That is, given the kth column of $\mathbf{Y}$, i.e y_k, we aim to find a subset of features, such that their linear span is close to y_k. This idea can be formulated as the minimization problem

$$\widehat{\Phi} = \underset{\Phi}{\operatorname{argmin}} \sum_{k=1}^{C} \|y_k - X\Phi_k\|^2 \ . \tag{11}$$

where $\Phi = [\Phi_1, \cdots, \Phi_k, \cdots, \Phi_C]$ and Φ_k is a d dimensional vector that contains the combination coefficients required to compute for different features in approximating y_k. However, feature selection requires to locate a optimal subset of features that are close to y_k. This is a combinatorial problem which is NP-hard. Thus we approximate the problem in (11) subject to the constraint

$$|\Phi_k| \leq \gamma \tag{12}$$

where $|\Phi_k|$ is the ℓ_1-norm and $|\Phi_k| = \sum_{j=1}^{d}|\Phi_{j,k}|$. When applied in regression, the ℓ_1-norm constraint is equivalent to applying a Laplace prior [18] on Φ_k. This tends to force some entries in Φ_k to be zero, resulting in a sparse solution. Therefore, the representation $\mathbf{Y}$ is generated by using only a small set of selected features in $\mathbf{X}$.

In order to efficiently solve the optimization problem in Equations (11) and (12), we use the Least Angle Regression (LARs) algorithm [20]. Instead of setting the parameter γ, LARs allow us to control the sparseness of Φ_k. This is done by specifying the cardinality of the number of nonzero subset of Φ_k, which is particularly convenient for feature selection.

We consider selecting m features from the d feature candidates. For a dataset containing C clusters, we can compute C selection vectors $\{\Phi_k\}_{k=1}^{C} \in R^d$. The cardinality of each Φ_k is m and each entry in Φ_k corresponds to a feature. Here, we use the following computationally effective method for selecting exactly m features based on the C selection vectors. For every feature j, we define the HG score for the feature as

$$HGscore(j) = \max_k |\Phi_{j,k}| \ . \tag{13}$$

where $\Phi_{j,k}$ is the jth element of vector Φ_k. We then sort the features in descending order according to their HG scores, and then select the top m features.

5 Feature Evaluation Indices

Our proposed semi-supervised feature selection method utilizes hypergraph based label regression and the Least Angle Regression (LARs) algorithm for semi-supervised feature selection. It involves applying label regression to embed the data into a new space and then uses LARs to select features that align well to the embedded data resulting from label regression. In order to examine the performance of our proposed method (referred to as the HG+Semi), we need to assess the data transformation obtained and its useful information content it. In view of this, we would like to measure the performance of our proposed algorithm using three different indices, namely, (1) **data transformation**, (2) **classification accuracy** and (3) **redundancy rate**. Assume **F** is the set of selected features, the redundancy rate can be defined as:

$$RED(F) = \frac{1}{m(m-1)} \sum_{f_i, f_j \in \mathbf{F}, i>j} \rho_{i,j} \ . \tag{14}$$

where $\rho_{i,j}$ returns the Pearson correlation score between two features f_i and f_j. The measurement assesses the averaged correlation among all feature pairs, and a large value indicates that many selected features are strongly correlated, and thus redundancy is expected to exist in **F**.

6 Experiments and Comparisons

Data sets: The data sets used to test the performance of our proposed algorithm are publicly available face-recognition benchmarks. Table. 1 summarizes the coverage and properties of the three data-sets. The original images were normalized (in scale and orientation) such that the two eyes were aligned at the same position. Then, the facial areas were cropped to give images for matching. In Fig. 2, we show the closely cropped images and these all contain facial structure. In our experiments, we only examine the performance of our proposed method with 50% data labeled.

Table 1. Summary of benchmark data sets

Data-set	Examples	Features	Classes
ORL	400	1024	40
CMU PIE	1428	1024	68
AR	130	2400	10

(a) ORL dataset

(b) CMU PIE dataset

(c) AR dataset

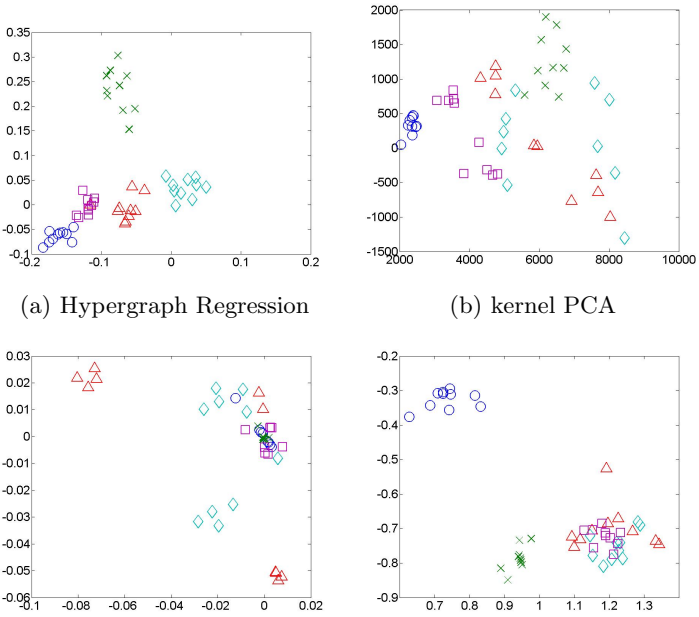
Fig. 2. The sample cropped face images form three face datasets

The ORL dataset contains 40 distinct individuals with ten images per person. The images are taken at different time instances, and include variations in facial expression and facial detail (glasses/no glasses). All images are resized to 32x32 pixels.

The CMU PIE dataset is a multiview face dataset, consisting of 41,368 images of 68 people. The views cover a wide range of poses from profile to frontal views, varying illumination and expression. In this experiment, we fixed the pose and expression. Thus, for each person, we have 21 images obtained under different lighting conditions. All images are resized to 32x32 pixels.

The AR dataset contains over 4000 face images from 126 people (70 men and 56 women). A subset of 10 subjects and 13 images per subject are selected in our experiments and the images are resized to 60×40 pixels. All the images are frontal view faces with different facial expressions, illumination conditions, and occlusions (sunglasses and scarf).

Data Transformation: we compare the data transformation performance of our proposed method using label regression with alternative methods, including kernel PCA [2], the Laplacian eigenmap [5] and LPP [3]. In order to visualize the results, we have used five randomly selected subjects from each dataset, and these are shown in Fig. 3, Fig. 4 and Fig. 5. In each figure, we have shown the projections onto the leading two most significant eigenmodes from different



(a) Hypergraph Regression (b) kernel PCA

(c) Laplacian eigenmaps (d) LPP

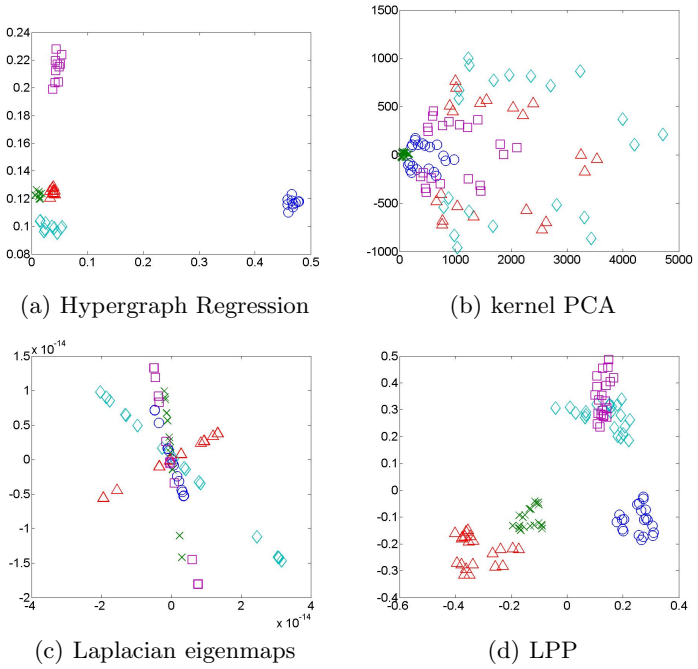
Fig. 3. Distribution of samples of five subjects in ORL dataset

spectral embedding methods, ordered according to their eigenvalues. This provides a low-dimensional representation for the images. From the above figures, it is clear that our hyypergraph based label regression method demonstrates much more clear cluster structure than that by traditional spectral clustering method. This implies that the hypergraph representation is both more appropriate and more complete in describing feature relations and structures.

Table 2. The best result of all methods and their corresponding size of selected feature subset on the three face datasets

Dataset	MRMR	FS	LS	SPEC	UDFS	HG+Semi
ORL	83.5%(95)	80%(99)	65.25%(99)	64.5%(95)	76.5%(99)	**93**%(**88**)
PIE	99.15%(99)	99.17%(100)	71.43%(99)	89.64%(100)	96%(98)	**99.19**%(**58**)
AR	88.15%(509)	87.69%(548)	60.77%(591)	86.15%(598)	82.31%(562)	**95.38**%(**427**)

Classification Accuracy: In order to explore the discriminative capabilities of the information captured by our method, we use the selected features for further classification. We compare the classification results from our proposed method HG+Semi with five representative feature selection algorithms. For unsupervised



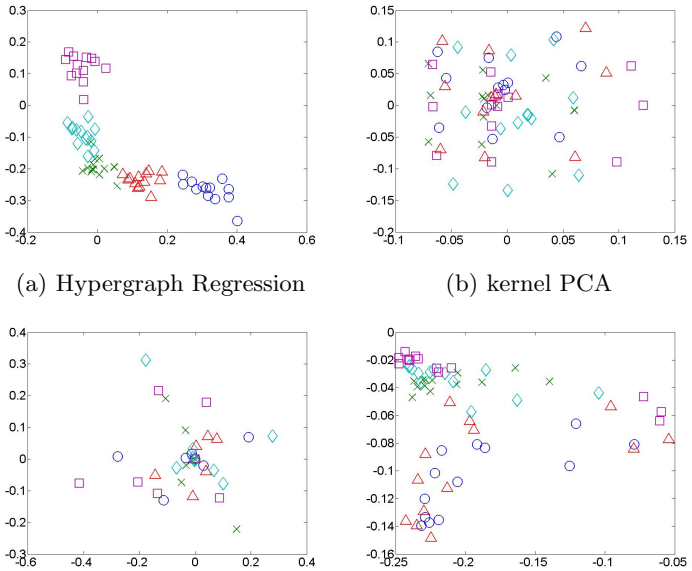
(a) Hypergraph Regression (b) kernel PCA

(c) Laplacian eigenmaps (d) LPP

Fig. 4. Distribution of samples of five subjects in CMU PIE dataset

learning, three alternative feature selection algorithms are selected as baselines. These methods are the Laplacian score (referred to as LS) [12], SPEC [11] and UDFS [23]. We also compare our results with two state-of-the-art supervised feature selection methods, namely a) the Fisher score (referred to as FS) [9] and b) the MRMR algorithm [7]. We use 5-fold cross-validation for the SVM classifier on the feature subsets obtained by the feature selection algorithms to verify their classification performance. Here we use the linear SVM with LIBSVM [21].

The classification accuracies obtained with different feature subsets are shown in Fig. 6. From the figure, it is clear that our proposed method HG+Semi is, by and large, superior to the alternative feature selection methods. Specifically, it selects both a smaller and better performing (in terms of classification accuracy) set of discriminative features on the three face data sets. Moreover, HG+Semi rapidly converges, with typically around 30 features (see Fig. 6(a)-(b)). Each of the alternative unsupervised methods, usually require more than 100 features to achieve a comparable result. There are two reasons for this improvement in performance. First, the hypergraph representation is effective in capturing the high-order relations among samples rather than approximating them in terms of pairwise interactions which can lead to a substantial loss of information. Thus the structural information latent in the data can be effectively preserved. Second, the LARs algorithm is applied to select features that align well to the embedded



(a) Hypergraph Regression (b) kernel PCA

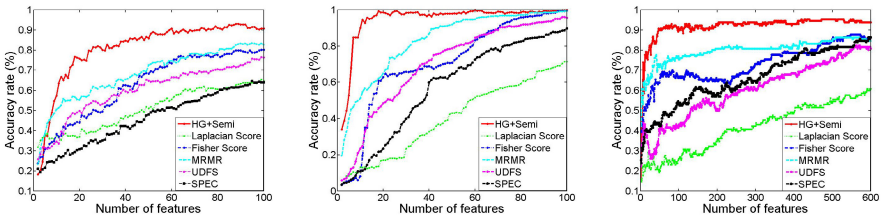
(c) Laplacian eigenmaps (d) LPP

Fig. 5. Distribution of samples of five subjects in AR dataset

data resulting from label regression. As a result the optimal feature combinations can be located so as to remove redundant features.

Compared with the two state-of-art supervised feature selection algorithms, our proposed semi-supervised method (HG+Semi) outperforms the MRMR algorithm and FS in all cases. On the CMU PIE dataset (see Fig. 6(b)), even though MRMR and FS can give good classification performance when more than 100 features are selected, HG+Semi achieves a comparable result with a much smaller number of features, i.e., less than 20 features. This implies that our proposed method is able to locate both the optimal size of the feature subset and perform accurate classification of the samples based on just a few of the most important features.

The best result for each method together with the corresponding size of the selected feature subset are shown in Table. 2. In the table, the classification accuracy is shown first and the optimal number of features selected is reported in brackets. Overall, HG+Semi achieves the highest degree of dimensionality reduction, i.e. it selects a smaller feature subset compared with those obtained by the alternative methods. For example, in the ORL data set, the best result obtained by the alternative feature selection methods is 83.5% with the MRMR algorithm and 95 features. However, our proposed method (HG+Semi) gives a better accuracy of 93% when only 88 features are used. The results further verify that our feature selection method can guarantee the optimal size of the feature subset, as it not only achieves a higher degree of dimensionality reduction but it



(a) ORL dataset (b) CMU PIE dataset (c) AR dataset

Fig. 6. Accuracy rate vs. the number of selected features on three face datasets

also gives better discriminability. We also observe that the UDFS algorithm gives a better result than the alternative unsupervised methods (i.e. the Laplacian score and the SPEC). The reason for this is that unlike traditional methods which treat each feature individually and which hence are suboptimal, the UDFS method directly optimizes the score over the entire selected feature subset. As a result, a better feature subset can be obtained.

Table 3. Averaged Redundancy rate of Subsets Selected Using Different Algorithms

Dataset	MRMR	FS	LS	SPEC	UDFS	HG+Semi
ORL	1.47	1.72	1.68	1.65	1.62	**1.38**
PIE	0.51	0.53	0.67	0.66	0.63	**0.45**
AR	0.56	0.68	0.76	0.70	0.72	**0.55**

Redundancy Rate: Table. 3 shows a comparison of results from our proposed method to the alternative feature selection methods using the top n features, where n is the number of training data. We chose n, since when the number of selected features is larger than n, any feature can be expressed by a linear combination of the remaining ones, which will introduce unnecessary redundancy in the evaluation stage. In the table, the boldfaced values are the lowest redundancy rates. The subset obtained by our proposed scheme has the least redundancy. This further verifies that our propose algorithm is able to remove redundant features.

The accuracy rate (Table. 2) and redundancy rate (Table. 3) together indicate that HG+Semi both gives the least redundancy, and results in the highest accuracy. They also underline the necessity of removing redundant features for improving learning performance. It should also be observed that the MRMR algorithm also produces low redundancy rates. However, it does not perform as well in terms of classification accuracy. This can be explained by the observation that in MRMR, feature contributions to the classification process are considered individually by evaluating the correlation between each feature and the class label. However, the class label may be jointly determined by a set of features. This interaction among features is not considered by MRMR.

7 Conclusion

In this paper, we have presented an semi-supervised feature selection method based on hypergraph subspace learning. The proposed feature selection method offers two major advantages. The first is that by incorporating MII for higher order similarities measure, we establish a novel hypergraph framework which is used for characterizing the multiple relationships within a set of samples. Thus, the structural information latent in the data can be more effectively modeled. Secondly, we derive a hypergraph subspace learning view of semi-supervised feature selection which casting the feature discriminant analysis into a regression framework that considers the correlations among features. As a result, we can evaluate joint feature combinations, rather than being confined to consider them individually. These properties enable our method to be able to handle feature redundancy effectively.

References

1. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. In: Proc. Syst. (1993)
2. Scholkopf, B., Smola, A., Muller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. In: Neural Computation, pp. 1299–1319 (1998)
3. He, X., Niyogi, P.: Locality preserving projections. Advances in Neural Information Processing Systems (2004)
4. He, X., Cai, D., Yan, S., Zhang, H.J.: Neighborhood preserving embedding. In: Tenth IEEE International Conference on Computer Vision, pp. 1208–1213 (2005)
5. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, pp. 585–592 (2002)
6. Battiti, R.: Using mutual information for selecting features in supervised neural net learning. IEEE Transactions on Neural Networks, 537–550 (2002)
7. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1226–1238 (2005)
8. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. Wiley, New York (2001)
9. Nie, F., Xiang, S., Jia, Y., Zhang, C., Yan, S.: Trace ratio criterion for feature selection. In: Proceedings of the 23rd National Conference on Artificial Intelligence, pp. 671–676 (2008)
10. Robnik-Šikonja, M., Kononenko, I.: Theoretical and empirical analysis of ReliefF and RReliefF. In: Machine Learning, pp. 23–69 (2003)
11. Zhao, Z., Liu, H.: Spectral feature selection for supervised and unsupervised learning. In: Proceedings of the 24th International Conference on Machine Learning, pp. 1151–1157 (2007)
12. He, X., Cai, D., Niyogi, P.: Laplacian score for feature selection. In: Advances in Neural Information Processing Systems (2005)
13. Belhumeur, P.N., Kriegman, D.J.: What is the set of images of an object under all possible illumination conditions? International Journal of Computer Vision, 245–260 (1998)

14. Agarwal, S., Branson, K., Belongie, S.: Higher order learning with graphs. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 17–24 (2006)
15. Agarwal, S., Lim, J., Zelnik-Manor, L., Perona, P., Kriegman, D., Belongie, S.: Beyond pairwise clustering. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 838–845 (2005)
16. Chung, F.: The Laplacian of a hypergraph. AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 21–36 (1993)
17. Li, W.C.W., Solé, P.: Spectra of regular graphs and hypergraphs and orthogonal polynomials. European Journal of Combinatorics, 461–477 (1996)
18. Seeger, M.W.: Bayesian inference and optimal design for the sparse linear model. The Journal of Machine Learning Research, 759–813 (2008)
19. Argyriou, A., Evgeniou, T., Pontil, M.: Convex multi-task feature learning. In: Machine Learning, pp. 243–272 (2008)
20. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. In: The Annals of Statistics, pp. 407–499 (2004)
21. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001)
22. Cai, D., Zhang, C., He, X.: Unsupervised feature selection for multi-cluster data. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 333–342 (2010)
23. Yang, Y., Shen, H.T., Ma, Z., Huang, Z., Zhou, X.: L21-norm regularized discriminative feature selection for unsupervised learning. In: International Joint Conferences on Artificial Intelligence, pp. 1589–1594 (2011)
24. Jacobs, D.W., Belhumeur, P.N., Basri, R.: Comparing images under variable illumination. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 610–617 (1998)
25. Nie, F., Xiang, S., Liu, Y., Zhang, C.: A general graph-based semi-supervised learning with novel class discovery. In: Neural Computing & Applications, pp. 549–555 (2010)
26. Zhang, Z., Hancock, E.R.: Feature selection for gender classification. In: 5th Iberian Conference on Pattern Recognition and Image Analysis, pp. 76–83 (2011)
27. Zhang, Z., Hancock, E.R.: Hypergraph based Information-theoretic Feature Selection. Pattern Recognition Letters (2012)
28. Yang, A.Y., Wright, J., Ma, Y., Sastry, S.S.: Feature selection in face recognition: A sparse representation perspective. IEEE Transactions on Pattern Analysis and Machine Intelligence (2007)
29. Nie, F., Xu, D., Li, X., Xiang, S.: Semi-supervised dimensionality reduction and classification through virtual label regression. IEEE Transactions on Systems, Man, and Cybernetics, 1–11 (2011)
30. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review, 455–500 (2009)
31. Shashua, A., Zass, R., Hazan, T.: Multi-way Clustering Using Super-Symmetric Non-negative Tensor Factorization. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 595–608. Springer, Heidelberg (2006)
32. Björck, A.: Numberical methods for least squares problems. In: Proc. SIAM (1996)

Learning Neighborhoods for Metric Learning

Jun Wang, Adam Woznica, and Alexandros Kalousis

AI Lab, Department of Computer Science, University of Geneva, Switzerland
Department of Business Informatics, University of Applied Sciences,
Western Switzerland
{jun.wang,adam.woznica}@unige.ch,
alexandros.kalousis@hesge.ch

Abstract. Metric learning methods have been shown to perform well on different learning tasks. Many of them rely on target neighborhood relationships that are computed in the original feature space and remain fixed throughout learning. As a result, the learned metric reflects the original neighborhood relations. We propose a novel formulation of the metric learning problem in which, in addition to the metric, the target neighborhood relations are also learned in a two-step iterative approach. The new formulation can be seen as a generalization of many existing metric learning methods. The formulation includes a target neighbor assignment rule that assigns different numbers of neighbors to instances according to their quality; 'high quality' instances get more neighbors. We experiment with two of its instantiations that correspond to the metric learning algorithms LMNN and MCML and compare it to other metric learning methods on a number of datasets. The experimental results show state-of-the-art performance and provide evidence that learning the neighborhood relations does improve predictive performance.

Keywords: Metric Learning, Neighborhood Learning.

1 Introduction

The choice of the appropriate distance metric plays an important role in distance-based algorithms such as k-NN and k-Means clustering. The Euclidean metric is often the metric of choice, however, it may easily decrease the performance of these algorithms since it relies on the simple assumption that all features are equally informative. Metric learning is an effective way to overcome this limitation by learning the importance of difference features exploiting prior knowledge that comes in different forms. The most well studied metric learning paradigm is that of learning the Mahalanobis metric with a steadily expanding literature over the last years [19,13,3,2,10,18,9,5,16].

Metric learning for classification relies on two interrelated concepts, similarity and dissimilarity constraints, and the target neighborhood. The latter defines for any given instance the instances that should be its neighbors and it is specified using similarity and dissimilarity constraints. In the absence of any other prior knowledge the similarity and dissimilarity constraints are derived from the class

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 223–236, 2012.

labels; instances of the same class should be similar and instances of different classes should be dissimilar.

The target neighborhood can be constructed in a *global* or *local* manner. With a global target neighborhood all constraints over all instance pairs are active; *all* instances of the same class should be similar and *all* instances from different classes should be dissimilar [19,3]. These admittedly hard to achieve constraints can be relaxed with the incorporation of slack variables [13,2,10,9]. With a local target neighborhood the satisfiability of the constraints is examined within a local neighborhood [4,17,10,18]. For any given instance we only need to ensure that we satisfy the constraints that involve that instance and instances from its local neighborhood. The resulting problem is considerably less constrained than what we get with the global approach and easier to solve. However, the appropriate definition of the local target neighborhood becomes now a critical component of the metric learning algorithm since it determines which constraints will be considered in the learning process. [18] defines the local target neighborhood of an instance as its k, same-class, nearest neighbors, under the Euclidean metric in the original space. Goldberger et al. [4] initialize the target neighborhood for each instance to all same-class instances. The local neighborhood is encoded as a soft-max function of a linear projection matrix and changes as a result of the metric learning. With the exception of [4], all other approaches whether global or local establish a target neighborhood prior to learning and keep it fixed throughout the learning process. Thus the metric that will be learned from these fixed neighborhood relations is constrained by them and will be a reflection of them. However, these relations are not necessarily optimal with respect to the learning problem that one is addressing.

In this paper we propose a novel formulation of the metric learning problem that includes in the learning process the learning of the local target neighborhood relations. The formulation is based on the fact that many metric learning algorithms can be seen as directly maximizing the sum of some quality measure of the target neighbor relationships under an explicit parametrization of the target neighborhoods. We cast the process of learning the neighborhood as a linear programming problem with a totally unimodular constraint matrix [14]. An integer 0-1 solution of the target neighbor relationship is guaranteed by the totally unimodular constraint matrix. The number of the target neighbors does not need to be fixed, the formulation allows the assignment of a different number of target neighbors for each learning instance according to the instance's quality. We propose a two-step iterative optimization algorithm that learns the target neighborhood relationships and the distance metric. The proposed neighborhood learning method can be coupled with standard metric learning methods to learn the distance metric, as long as these can be cast as instances of our formulation.

We experiment with two instantiations of our approach where the Large Margin Nearest Neighbor (LMNN) [18] and Maximally Collapsing Metric Learning (MCML) [3] algorithms are used to learn the metric; we dub the respective instantiations LN-LMNN and LN-MCML. We performed a series of experiments on a number of classification problems in order to determine whether learning

the neighborhood relations improves over only learning the distance metric. The experimental results show that this is indeed the case. In addition, we also compared our method with other state-of-the-art metric learning methods and show that it improves over the current state-of-the-art performance.

The paper is organized as follows. In section 2, we discuss in more detail the related work. In Section 3 we present the optimization problem of the Learning Neighborhoods for Metric Learning algorithm (LNML) and in Section 4 we discuss the properties of LNML. In Section 5 we instantiate our neighborhood learning method on LMNN and MCML. In Section 6 we present the experimental results and we finally conclude with Section 7.

2 Related Work

The early work of Xing et al., [19], learns a Mahalanobis distance metric for clustering that tries to minimize the sum of pairwise distances between similar instances while keeping the sum of dissimilar instance distances greater than a threshold. The similar and dissimilar pairs are determined on the basis of prior knowledge. Globerson & Roweis, [3] introduced the Maximally Collapsing Metric Learning (MCML). MCML uses a stochastic nearest neighbor selection rule which selects the nearest neighbor $\mathbf{x}_j$ of an instance $\mathbf{x}_i$ under some probability distribution. It casts the metric learning problem as an optimization problem that tries to minimize the distance between two probability distributions, an ideal one and a data dependent one. In the ideal distribution the selection probability is always one for instances of the same class and zero for instances of different class, defining in that manner the similarity and dissimilarity constraints under the global target neighborhood approach. In the data dependent distribution the selection probability is given by a soft max function of a Mahalanobis distance metric, parametrized by the matrix $\mathbf{M}$ to be learned. In a similar spirit Davis et al., [2], introduced Information-Theoretic Metric Learning. ITML learns a Mahalanobis metric for classification with similarities and dissimilarities constraints that follow the global target neighborhood approach. In ITML all same-class instance pairs should have a distance smaller than some threshold and all different-class instance pairs should have a distance larger than some threshold. In addition the objective function of ITML seeks to minimize the distance between the learned metric matrix and a prior metric matrix, modelling like that prior knowledge about the metric if such is available. The optimization problem is cast as a distance of distributions subject to the pairwise constraints and finally expressed as a Bregman optimization problem (minimizing the LogDet divergence). In order to be able to find a feasible solution they introduce slack variables in the similarity and dissimilarity constraints.

The so far discussed metric learning methods follow the global target neighborhood approach in which all instances of the same class should be similar under the learned metric, and all pairs of instances from different classes dissimilar. This is a rather hard constraint and assumes that there is a linear projection of the original feature space that results in unimodal class conditional distributions. Goldberger et al., [4], proposed the NCA metric learning method which

uses the same stochastic nearest neighbor selection rule under the same data-dependent probability distribution as MCML. NCA seeks to minimize the soft error under its stochastic nearest neighbor selection rule. It uses only similarity constraints and the original target neighborhood of an instance is the set of all same-class instances. After metric learning some, but not necessarily all, same class instances will end up having high probability of been selecting as nearest neighbors of a given instance, thus having a small distance, while the others will be pushed further away. NCA thus learns the local target neighborhood as a part of the optimization. Nevertheless it is prone to overfitting, [20], and does not scale to large datasets. The large margin nearest neighbor method (LMNN) described in [17,18] learns a distance metric which directly minimizes the distances of each instance to its local target neighbors while keeping a large margin between them and different class instances. The target neighbors have to be specified prior to metric learning and in the absence of prior knowledge these are the k same class nearest neighbors for each instance.

3 Learning Target Neighborhoods for Metric Learning

Given a set of training instances $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \ , \ldots, (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ and the class labels $y_i \in \{1, 2, \ldots, c\}$, the Mahalanobis distance between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as:

$$D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j) \tag{1}$$

where $\mathbf{M}$ is a Positive Semi-Definite (PSD) matrix ($\mathbf{M} \succeq 0$) that we will learn.

We can reformulate many of the existing metric learning methods, such as [19,13,3,10,18], by explicitly parametrizing the target neighborhood relations as follows:

$$\begin{aligned} &\min_{\mathbf{M}, \boldsymbol{\Xi}} \sum_{ij, i \neq j, y_i = y_j} \mathbf{P}_{ij} \cdot F_{ij}(\mathbf{M}, \boldsymbol{\Xi}) \\ &s.t. \quad \text{constraints of the original problem} \end{aligned} \tag{2}$$

The matrix $\mathbf{P}, \mathbf{P}_{ij} \in \{0, 1\}$, describes the target neighbor relationships which are established prior to metric learning and are not altered in these methods. $\mathbf{P}_{ij} = 1$, if $\mathbf{x}_j$ is the target neighbor of $\mathbf{x}_i$, otherwise, $\mathbf{P}_{ij} = 0$. Note that the parameters $\mathbf{P}_{ii}$ and $\mathbf{P}_{ij} : y_i \neq y_j$ are set to zero, since an instance $\mathbf{x}_i$ cannot be a target neighbor of itself and the target neighbor relationship is constrained to same-class instances. This is why we have $i \neq j, y_i = y_j$ in the sum, however, for simplicity we will drop it from the following equations. $F_{ij}(\mathbf{M}, \boldsymbol{\Xi})$ is the term of the objective function of the metric learning methods that relates to the target neighbor relationship $\mathbf{P}_{ij}$, $\mathbf{M}$ is the Mahalanobis metric that we want to learn, and $\boldsymbol{\Xi}$ is a set of other parameters in the original metric learning problems, e.g. slack variables. Regularization terms on the $\mathbf{M}$ and $\boldsymbol{\Xi}$ parameters can also be added into Problem 2 [13,10].

The $F_{ij}(\mathbf{M}, \boldsymbol{\Xi})$ term can be seen as the 'quality' of the target neighbor relationship $\mathbf{P}_{ij}$ under the distance metric $\mathbf{M}$; a low value indicates a high quality

neighbor relationship $\mathbf{P}_{ij}$. The different metric learning methods learn the $\mathbf{M}$ matrix that optimizes the sum of the quality terms based on the a priori established target neighbor relationships; however, there is no reason to believe that these target relationships are the most appropriate for learning.

To overcome the constraints imposed by the fixed target neighbors we propose the Learning the Neighborhood for Metric Learning method (LNML) in which, in addition to the metric matrix $\mathbf{M}$, we also learn the target neighborhood matrix $\mathbf{P}$. LNML has as objective function the one given in Problem 2 which we now optimize also over the target neighborhood matrix $\mathbf{P}$. We add some new constraints in Problem 2 which control for the size of the target neighborhoods. The new optimization problem is the following:

$$\begin{aligned} \min_{\mathbf{M},\Xi,\mathbf{P}} \quad & \sum_{ij} \mathbf{P}_{ij} \cdot F_{ij}(\mathbf{M}, \Xi) \\ s.t. \quad & \sum_{i,j} \mathbf{P}_{ij} = K_{av} * n \\ & K_{max} \geq \sum_{j} \mathbf{P}_{i,j} \geq K_{min} \\ & 1 \geq \mathbf{P}_{ij} \geq 0 \\ & \text{constraints of the original problem} \end{aligned} \tag{3}$$

K_{min} and K_{max} are the minimum and maximum numbers of target neighbors that an instance can have. Thus the second constraint controls the number of target neighbor that $\mathbf{x}_i$ instance can have. K_{av} is the average number of target neighbor per instance. It holds by construction that $K_{max} \geq K_{av} \geq K_{min}$. We should note here that we relax the target neighborhood matrix so that its elements $\mathbf{P}_{ij}$ take values in $[0, 1]$ (third constraint). However, we will show later that a solution $\mathbf{P}_{ij} \in \{0, 1\}$ is obtained, given some natural constraints on the K_{min}, K_{max} and K_{av} parameters.

3.1 Target Neighbor Assignment Rule

Unlike other metric learning methods, e.g. LMNN, in which the number of target neighbors is fixed, LNML can assign a different number of target neighbors for each instance. As we saw the first constraint in Problem 3 sets the average number of target neighbors per instance to K_{av}, while the second constraint limits the number of target neighbors for each instance between K_{min} and K_{max}. The above optimization problem implements a target neighbor assignment rule which assigns more target neighbors to instances that have high quality target neighbor relations. We do so in order to avoid overfitting since most often the 'good' quality instances defined by metric learning algorithms [3,18] are instances in dense areas with low classification error. As a result the geometry of the dense areas of the different classes will be emphasized. How much emphasis we give on good quality instances depends on the actual values of K_{min} and K_{max}. In the limit one can set the value of K_{min} to zero; nevertheless the risk with such

a strategy is to focus heavily on dense and easy to learn regions of the data and ignore important boundary instances that are useful for learning.

4 Optimization

4.1 Properties of the Optimization Problem

We will now show that we get integer solutions for the $\mathbf{P}$ matrix by solving a linear programming problem and analyze the properties of Problem 3.

Lemma 1. *Given* $\mathbf{M}, \boldsymbol{\Xi}$*, and* $K_{max} \geq K_{av} \geq K_{min}$ *then* $\mathbf{P}_{ij} \in \{0, 1\}$*, if* K_{min}*,* K_{max} *and* K_{av} *are integers.*

Proof. Given $\mathbf{M}$ and $\boldsymbol{\Xi}$, $F_{ij}(\mathbf{M}, \boldsymbol{\Xi})$ becomes a constant. We denote by $\mathbf{p}$ the vectorization of the target neighborhood matrix $\mathbf{P}$ which excludes the diagonal elements and $\mathbf{P}_{ij} : y_i \neq y_j$, and by $\mathbf{f}$ the respective vectorized version of the F_{ij} terms. Then we rewrite Problem 3 as:

$$\begin{aligned} \min_{\mathbf{p}} \ & \mathbf{p}^T \mathbf{f} \\ s.t. \ & (\underbrace{K_{max}, \cdots, K_{max}}_{n}, K_{av} * n)^T \geq \mathbf{A}\mathbf{p} \geq \\ & (\underbrace{K_{min}, \cdots, K_{min}}_{n}, K_{av} * n)^T \\ & 1 \geq \mathbf{p}_i \geq 0 \end{aligned} \tag{4}$$

The first and second constraints of Problem 3 are reformulated as the first constraint in Problem 4. $\mathbf{A}$ is a $(n+1) \times (\sum_{c_l} n_{c_l}^2 - n)$ constraint matrix, where n_{c_l} is the number of instances in class c_l

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \cdots & \mathbf{1} \end{bmatrix}$$

where $\mathbf{1}$ ($\mathbf{0}$) is the vector of ones (zeros). Its elements depends on the its position in the matrix $\mathbf{A}$. In its ith column, all $\mathbf{1}$ ($\mathbf{0}$) vectors have $n_i - 1$ elements, where n_i is the number of instances of class c_j with $c_j = y_{p_i}$. According to the sufficient condition for total unimodularity (Theorem 7.3 in [14]) the constraint matrix $\mathbf{A}$ is a totally unimodular matrix. Thus, the constraint matrix $\mathbf{B} = [\mathbf{I}, -\mathbf{I}, \mathbf{A}, -\mathbf{A}]^T$ in the following equivalent problem also is a totally unimodular matrix (pp.268 in [12]).

$$\begin{aligned} \min_{\mathbf{p}} \ & \mathbf{p}^T\mathbf{f} \\ s.t. \ & \mathbf{Bp} \leq \mathbf{e} \\ & e = (\underbrace{1,\cdots,1}_{\sum_{c_l} n_{c_l}^2 - n}, \underbrace{0,\cdots,0}_{\sum_{c_l} n_{c_l}^2 - n}, \underbrace{K_{max},\cdots,K_{max}}_{n}, \\ & K_{av} * n, \underbrace{-K_{min},\cdots,-K_{min}}_{n}, -K_{av} * n)^T \end{aligned} \tag{5}$$

Since $\mathbf{e}$ is an integer vector, provided K_{min}, K_{max}, and K_{av}, are integers, and the constraint matrix $\mathbf{B}$ is totally unimodular, the above linear programming problem will only have integer solutions (Theorem 19.1a in [12]). Therefore, for the solution $\mathbf{p}$ it will hold that $\mathbf{p}_i \in \{0, 1\}$ and consequently $\mathbf{P}_{ij} \in \{0, 1\}$.

Although the constraints to control the size of the target neighborhood are convex, the objective function in Problem 3 is not jointly convex in $\mathbf{P}$ and $(\mathbf{M}, \boldsymbol{\Xi})$. However, as shown in Lemma 1, the binary solution of $\mathbf{P}$ can be obtained by a simple linear program if we fix $(\mathbf{M}, \boldsymbol{\Xi})$. Thus, Problem 3 is individually convex in $\mathbf{P}$ and $(\mathbf{M}, \boldsymbol{\Xi})$, if the original metric learning method is convex; this condition holds for all the methods that can be coupled with our neighborhood learning method [19,13,3,10,18].

4.2 Optimization Algorithm

Based on Lemma 1 and the individual convexity property we propose a general and easy to implement iterative algorithm to solve Problem 3. The details are given in Algorithm 1. At the first step of the kth iteration we learn the binary target neighborhood matrix $\mathbf{P}^{(k)}$ under a fixed metric matrix $\mathbf{M}^{(k-1)}$ and $\boldsymbol{\Xi}^{(k-1)}$, learned in the $k-1$th iteration, by solving the linear programming problem described in Lemma 1. At the second step of the iteration we learn the metric matrix $\mathbf{M}^{(k)}$ and $\boldsymbol{\Xi}^{(k)}$ with the target neighborhood matrix $\mathbf{P}^{(k)}$ using as the initial metric matrix the $\mathbf{M}^{(k-1)}$. The second step is simply the application of a standard metric learning algorithm in which we set the target neighborhood matrix to the learned $\mathbf{P}^{(k)}$ and the initial metric matrix to $\mathbf{M}^{(k-1)}$. The convergence of proposed algorithm is guaranteed if the original metric learning problem is convex [1]. In our experiment, it most often converges in 5-10 iterations.

5 Instantiating LNML

In this section we will show how we instantiate our neighborhood learning method with two standard metric learning methods, LMNN and MCML, other possible instantiations include the metric learning methods presented in [19,13,10].

Algorithm 1. LNML

Input: $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{M}^0$,$\mathbf{\Xi}^0$, K_{min}, K_{max} and K_{av}
Output: $\mathbf{M}$
repeat
$\mathbf{P}^{(k)}$=LearningNeighborhood($\mathbf{X}, \mathbf{Y}, \mathbf{M}^{(k-1)}, \mathbf{\Xi}^{(k-1)}$) by solving Problem 4
$(\mathbf{M}^{(k)}, \mathbf{\Xi}^{(k)})$=MetricLearning($\mathbf{M}^{(k-1)}$,$\mathbf{P}^{(k)}$)
$k := k + 1$
until convergence

5.1 Learning the Neighborhood for LMNN

The optimization problem of LMNN is given by:

$$\begin{aligned} \min_{\mathbf{M},\xi} \ & \sum_{ij} \mathbf{P}_{ij}\{(1-\mu)D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) + \mu \sum_{l}(1-\mathbf{Y}_{il})\xi_{ijl}\} \\ s.t. \ & D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_l) - D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 - \xi_{ijl} \\ & \xi_{ijl} > 0 \\ & \mathbf{M} \succeq 0 \end{aligned} \tag{6}$$

where the matrix $\mathbf{Y}, \mathbf{Y}_{ij} \in \{0,1\}$, indicates whether the class labels y_i and y_j are the same ($\mathbf{Y}_{ij} = 1$) or different ($\mathbf{Y}_{ij} = 0$). The objective is to minimize the sum of the distances of all instances to their target neighbors while allowing for some errors, this trade off is controlled by the μ parameter. This is a convex optimization problem that has been shown to have good generalization ability and can be applied to large datasets. The original problem formulation corresponds to a fixed parametrization of $\mathbf{P}$ where its non-zero values are given by the k nearest neighbors of the same class.

Coupling the neighborhood learning framework with the LMNN metric learning method results in the following optimization problem:

$$\begin{aligned} & \min_{\mathbf{M},\mathbf{P},\xi} \ \sum_{ij} \mathbf{P}_{ij} \cdot F_{ij}(\mathbf{M},\xi) \\ = & \min_{\mathbf{M},\mathbf{P},\xi} \ \sum_{ij} \mathbf{P}_{ij}\{(1-\mu)D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) + \mu \sum_{l}(1-\mathbf{Y}_{il})\xi_{ijl}\} \\ & s.t. \ K_{max} \geq \sum_{j} \mathbf{P}_{i,j} \geq K_{min} \\ & \sum_{i,j} \mathbf{P}_{ij} = K_{av} * n \\ & 1 \geq \mathbf{P}_{ij} \geq 0 \\ & D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_l) - D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 - \xi_{ijl} \\ & \xi_{ijl} > 0 \\ & \mathbf{M} \succeq 0 \end{aligned} \tag{7}$$

We will call this coupling of LNML and LMNN LN-LMNN. The target neighbor assignment rule of LN-LMNN assigns more target neighbors to instances that have small distances from their target neighbors and low hinge loss.

5.2 Learning the Neighborhood for MCML

MCML relies on a data dependent stochastic probability that an instance $\mathbf{x}_j$ is selected as the nearest neighbor of an instance $\mathbf{x}_i$; this probability is given by:

$$p_{\mathbf{M}}(j|i) = \frac{e^{-D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)}}{Z_i} = \frac{e^{-D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)}}{\sum_{k\neq i} e^{-D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k)}}, i \neq j \tag{8}$$

MCML learns the Mahalanobis metric that minimizes the KL divergence distance between this probability distribution and the ideal probability distribution p_0 given by:

$$p_0(j|i) = \frac{\mathbf{P}_{ij}}{\sum_k \mathbf{P}_{ik}}, p_0(i|i) = 0 \tag{9}$$

where $\mathbf{P}_{ij} = 1$, if instance $\mathbf{x}_j$ is the target neighbor of instance $\mathbf{x}_i$, otherwise, $\mathbf{P}_{ij} = 0$. The optimization problem of MCML is given by:

$$\begin{aligned} &\min_{\mathbf{M}} \sum_i KL[p_0(j|i)|p_{\mathbf{M}}(j|i)] \\ &= \min_{\mathbf{M}} \sum_{i,j} \mathbf{P}_{ij} \frac{(D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) + \log Z_i)}{\sum_k \mathbf{P}_{ik}} \\ &s.t.\ \mathbf{M} \succeq 0 \end{aligned} \tag{10}$$

Like LMNN, this is also a convex optimization problem. In the original problem formulation the ideal distribution is defined based on class labels, i.e. $\mathbf{P}_{ij} = 1$, if instances $\mathbf{x}_i$ and $\mathbf{x}_j$ share the same class label, otherwise, $\mathbf{P}_{ij} = 0$.

The neighborhood learning method cannot learn directly the target neighborhood for MCML, since the objective function of the latter cannot be rewritten in the form of the objective function in Problem 3, due to the denominator $\sum_k \mathbf{P}_{ik}$. However, if we fix the size of the neighborhood to $\sum_k \mathbf{P}_{i,k} = K_{av} = K_{min} = K_{max}$ the two methods can be coupled and the resulting optimization is given by:

$$\begin{aligned} &\min_{\mathbf{M},\mathbf{P}} \sum_{ij} \mathbf{P}_{ij} \cdot F_{ij}(\mathbf{M}) \\ &= \min_{\mathbf{M},\mathbf{P}} \sum_{i,j} \mathbf{P}_{ij} \frac{(D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) + \log Z_i)}{K_{av}} \\ &s.t.\ \sum_j \mathbf{P}_{i,j} = K_{av} \\ &\mathbf{M} \succeq 0 \end{aligned} \tag{11}$$

We will dub this coupling of LNML and MCML as LN-MCML. The original MCML method follows the global approach in establishing the neighborhood, with LN-MCML we get a local approach in which the neighborhoods are of fixed size K_{av} for every instance.

6 Experiments

With the experiments we wish to investigate a number of issues. First, we want to examine whether learning the target neighborhood relations in the metric learning process can improve predictive performance over the baseline approach of metric learning with an apriori established target neighborhood. Second, we want to acquire an initial understanding of how the parameters K_{min} and K_{max} relate to the predictive performance. To this end, we will examine the predictive performance of LN-LMNN with two fold inner Cross Validation (CV) to select the appropriate values of K_{min} and K_{max}, method which we will denote by LN-LMNN(CV), and that of LN-LMNN, with a default setting of $K_{min} = K_{max} = K_{av}$. Finally, we want to see how the method that we propose compares to other state of the art metric learning methods, namely NCA and ITML. We include as an additional baseline in our experiments the performance of the Euclidean metric (EucMetric). We experimented with twelve different datasets: seven from the UCI machine learning repository, Sonar, Ionosphere, Iris, Balance, Wine, Letter, Isolet; four text mining datasets, Function, Alt, Disease and Structure, which were constructed from biological corpora [7]; and MNIST [8], a handwritten digit recognition problem. A more detailed description of the datasets is given in Table 1.

Since LMNN is computationally expensive for datasets with large number of features we applied principal component analysis (PCA) to retain a limited number of principal components, following [18]. The datasets to which this was done were the four text mining datasets, Isolet and MNIST. For the two latter 173 and 164 principal components were respectively retained that explain 95% of the total variance. For the text mining datasets more than 1300 principal components should be retained to explain 95% of the total variance. Considering the running time constraints, we kept the 300 most important principal components which explained 52.45%, 47.57%, 44.30% and 48.16% of the total variance for respectively Alt, Disease, Function and Structure. We could experiment with NCA and MCML on full tranining datasets only with datasets with a small number of instances due to their computational complexity. For completeness we experimented with NCA on large datasets by undersampling the training instances, i.e. the learning process only involved 10% of full training instances which was the maximum number we could experiment for each dataset. We also applied ITML on both versions of the larger datasets, i.e. with PCA-based dimensionality reduction and the original ones.

For ITML, we randomly generate for each dataset the default $20c^2$ constraints which are bounded repectively by the 5th and 95th percentiles of the distribution of all available pairwise distances for similar and dissimilar pairs. The slack

Table 1. Datasets

Datasets	Description	# Sample	# Feature	# Class	# Retained PCA Components	% Explained Variance
Sonar		208	60	2	NA	NA
Ionosphere		351	34	2	NA	NA
Wine		178	13	3	NA	NA
Iris		150	4	3	NA	NA
Balance		625	4	3	NA	NA
Letter	character recognition	20000	16	26	NA	NA
Function	sentence classification	3907	2708	2	300	44.30%
Alt	sentence classification	4157	2112	2	300	52.45%
Disease	sentence classification	3273	2376	2	300	47.57%
Structure	sentence classification	3584	2368	2	300	48.16%
Isolet	spoken character recognition	7797	619	26	173	95%
MNIST	handwritten digit recognition	70000	784	26	164	95%

variable γ is chosen form $\{10^i\}_{i=-4}^{4}$ using two-fold CV. The default identity matrix is employed as the regularization matrix. For the different instantiations of the LNML method we took care to have the same parameter settings for the encapsulated metric learning method and the respective baseline metric learning. For LN-LMNN, LN-LMNN(CV) and LMNN the regularization parameter μ that controls the trade-off between the distance minimization component and the hinge loss component was set to 0.5 (the default value of LMNN). For LMNN the default number of target neighbors was used (three). For LN-LMNN, we set $K_{min} = K_{max} = K_{av} = 3$, similar to LMNN. To explore the effect of a flexible neighborhood, the values of the K_{min} and K_{max} parameters in LN-LMNN(CV) were selected from the sets $\{1, 4, 3\}$ and $\{2, 5, 3\}$ respectively, while K_{av} was fixed to three. Similarly for LN-MCML we also set $K_{av} = 3$. The distance metrics for all methods are initialized to the Euclidean metric. As the classification algorithm we used 1-Nearest Neighbor.

We used 10-fold cross validation for all datasets to estimate classification accuracy, with the exception of Isolet and MNIST for which the default train and test split was used. The statistical significance of the differences were tested with McNemar's test and the p-value was set to 0.05. In order to get a better understanding of the relative performance of the different algorithms for a given dataset we used a ranking schema in which an algorithm A was assigned one point if it was found to have a significantly better accuracy than another algorithm B, 0.5 points if the two algorithms did not have a significantly different performance, and zero points if A was found to be significantly worse than B. The rank of an algorithm for a given dataset is simply the sum of the points over the different pairwise comparisons. When comparing N algorithms in a single dataset the highest possible score is $N - 1$ while if there is no significant difference each algorithm will get $(N - 1)/2$ points.

6.1 Results

The results are presented in Table 2. Examining whether learning also the neighborhood improves the predictive performance compared to plain metric learning, we see that in the case of LN-MCML, and for the five small datasets for which we have results, learning the neighborhood results in a statistically significant

deterioration of the accuracy in one out of the five datasets (balance), while for the remaining four the differences were not statistically significant. If we now examine LN-LMNN(CV), LN-LMNN and LMNN we see that here learning the neighborhood does bring a statistically significant improvement. More precisely, LN-LMNN(CV) and LN-LMNN improve over LMNN respectively in six (two small and four large) and four (two small and two large) out of the 12 datasets. Moreove, by comparing LN-LMNN(CV) and LN-LMNN, we see that learning a flexible neighborhood with LN-LMNN(CV) improves significantly the performance over LN-LMNN on two datasets. The low performance of LN-MCML on the balance dataset was intriguing; in order to take a closer look we tried to determine automatically the appropriate target neighborhood size, K_{av}, by selecting it on the basis of five-fold inner cross validation from the set $K_{av} = \{3, 5, 7, 10, 20, 30\}$. The results showed that the default value of K_{av} was too small for the given dataset, with the average selected size of the target neighborhood at 29. As a result of the automatic tunning of the target neighborhood size the predictive performance of LN-MCML jumped at an accuracy of 93.92% which represented a significant improvement over the baseline MCML for the balance dataset. For the remaining datasets it turned out that the choice of $K_{av} = 3$ was a good default choice. In any case, determining the appropriate size of the target neighborhood and how that affects the predictive performance is an issue that we wish to investigate further. In terms of the total score that the different methods obtain the LN-LMNN(CV) achieves the best in both the

Table 2. Accuracy results. The superscripts $^{+-=}$ next to the LN-XXXX accuracy indicate the result of the McNemar's statistical test result of its comparison to the accuracy of XXXX and denote respectively a significant win, loss or no difference for LN-XXXX. Similarly, the superscripts $^{+-=}$ next to the LN-LMNN(CV) accuracy indicate the result of its comparison to the accuracies of LMNN and LN-LMNN. The bold entries for each dataset have no significant difference from the best accuracy for that dataset. The number in the parenthesis indicates the score of the respective algorithm for the given dataset based on the pairwise comparisons.

(a) Small datasets

Datasets	MCML	LN-MCML	LMNN	LN-LMNN	LN-LMNN(CV)	EucMetric	NCA	ITML
Sonar	**82.69**(3.5)	**84.62**$(3.5)^{=}$	**81.25**(3.5)	**81.25**$(3.5)^{=}$	**83.17**$(3.5)^{==}$	**80.77**(3.5)	**81.73**(3.5)	**82.69**(3.5)
Ionosphere	88.03 (3.0)	**88.89**$(3.5)^{=}$	**89.17**(3.5)	87.75 $(3.0)^{=}$	**92.02**$(5.5)^{=+}$	86.32 (3.0)	**88.60**(3.5)	87.75 (3.0)
Wine	91.57 (3.0)	**96.07**$(4.0)^{=}$	94.38 (3.0)	**97.75**$(5.5)^{+}$	**97.75**$(5.5)^{+=}$	76.97 (0.0)	91.57 (3.0)	**94.94**(4.0)
Iris	**98.00**(4.5)	**96.00**$(3.5)^{=}$	**96.00**(3.5)	94.00 $(3.0)^{=}$	94.00 $(3.0)^{==}$	**96.00**(3.5)	**96.00**(3.5)	**96.00**(3.5)
Balance	91.20 (5.0)	78.08 $(1.0)^{-}$	78.56 (1.0)	89.12 $(4.5)^{+}$	89.28 $(4.5)^{+=}$	78.72 (1.0)	**96.32**(7.0)	87.84 (4.0)
Total Score	19.0	15.5	14.5	19.5	22.0	11.0	20.5	18.0

(b) Large datasets

Datasets	PCA+LMNN	PCA+LN-LMNN	PCA+LN-LMNN(CV)	EucMetric	PCA+EucMetric	PCA+NCA	ITML	PCA+ITML
Letter	96.86 (5.0)	**97.71**$(6.5)^{+}$	**97.64**$(6.5)^{+=}$	96.02 (0.5)	96.02 (0.5)	96.48 (3.0)	96.39 (3.0)	96.39 (3.0)
Function	76.30 (2.5)	76.73 $(2.5)^{=}$	**78.91**$(6.0)^{++}$	**78.73**(6.0)	76.48 (2.5)	72.36 (0.0)	**78.73**(6.0)	76.45 (2.5)
Alt	83.98 (5.0)	**84.92**$(6.5)^{+}$	**85.37**$(6.5)^{+=}$	68.51 (0.5)	71.33 (2.0)	78.54 (4.0)	68.49 (0.5)	72.53 (3.0)
Disease	**80.23**(4.0)	**80.14**$(4.0)^{=}$	**80.66**$(4.0)^{==}$	**80.60**(4.0)	**80.23**(4.0)	73.59 (0.0)	**80.60**(4.0)	**80.14**(4.0)
Structure	77.87 (4.5)	**78.83**$(6.0)^{=}$	**79.37**$(6.5)^{+=}$	75.82 (1.5)	77.00 (4.0)	71.93 (0.0)	75.79 (1.5)	77.06 (4.0)
Isolet	**95.96**(6.0)	**95.06**$(6.0)^{=}$	**95.06**$(6.0)^{==}$	88.58 (1.5)	88.33 (1.5)	85.63(0.0)	92.05 (3.5)	91.08 (3.5)
MNIST	**97.66**(6.0)	**97.66**$(6.0)^{=}$	**97.73**$(6.0)^{==}$	96.91 (2.0)	96.97 (2.0)	96.58 (1.5)	96.93 (1.5)	97.09 (3.0)
Total Score	33	37.5	41.5	16	16.5	8.5	20	23

small and large datasets. It is followed closely by NCA in the small datasets and by LN-LMNN in the large datasets.

7 Conclusion and Future Work

We presented LNML, a general Learning Neighborhood method for Metric Learning algorithms which couples the metric learning process with the process of establishing the appropriate target neighborhood for each instance, i.e. discovering for each instance which same class instances should be its neighbors. With the exception of NCA, which cannot be applied on datasets with many instances, all other metric learning methods whether they establish a global or a local target neighborhood do that prior to the metric learning and keep the target neighborhood fixed throughout the learning process. The metric that is learned as a result of the fixed neighborhoods simply reflects these original relations which are not necessarily optimal with respect to the classification problem that one is trying to solve. LNML lifts these constraints by learning the target neighborhood. We demonstrated it with two metric learning methods, LMNN and MCML. The experimental results show that learning the neighborhood can indeed improve the predictive performance.

The target neighborhood matrix $\mathbf{P}$ is strongly related to the similarity graphs which are often used in semi-supervised learning [6], spectral clustering [15] and manifold learning [11]. Most often the similarity graphs in these methods are constructed in the original space, which nevertheless can be quite different from true manifold on which the data lies. These methods could also profit if one is able to learn the similarity graph instead of basing it on some prior structure.

Acknowledgments. This work was funded by the Swiss NSF (Grant 200021-122283/1). The support of the European Commission through EU projects DebugIT (FP7-217139) and e-LICO (FP7-231519) is also gratefully acknowledged.

References

1. Bezdek, J.C., Hathaway, R.J.: Some Notes on Alternating Optimization. In: Pal, N.R., Sugeno, M. (eds.) AFSS 2002. LNCS (LNAI), vol. 2275, pp. 288–300. Springer, Heidelberg (2002)
2. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: Proceedings of the 24th International Conference on Machine Learning. ACM, New York (2007)
3. Globerson, A., Roweis, S.: Metric learning by collapsing classes. In: Advances in Neural Information Processing Systems, vol. 18, MIT Press (2006)
4. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. In: Advances in Neural Information Processing Systems, vol. 17, MIT Press (2005)
5. Guillaumin, M., Verbeek, J., Schmid, C.: Is that you? Metric learning approaches for face identification. In: Proceedings of 12th International Conference on Computer Vision, pp. 498–505 (2009)

6. Jebara, T., Wang, J., Chang, S.-F.: Graph construction and b-matching for semi-supervised learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 441–448. ACM, New York (2009)
7. Kalousis, A., Prados, J., Hilario, M.: Stability of feature selection algorithms: a study on high-dimensional spaces. Knowledge and Information Systems 12(1), 95–116 (2007)
8. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324 (1998)
9. Lu, Z., Jain, P., Dhillon, I.S.: Geometry-aware metric learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM Press, New York (2009)
10. Nguyen, N., Guo, Y.: Metric Learning: A Support Vector Approach. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 125–136. Springer, Heidelberg (2008)
11. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science 22, 2323–2326 (2000)
12. Schrijver, A.: Theory of linear and integer programming. John Wiley & Sons Inc. (1998)
13. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference, p. 41. MIT Press (2004)
14. Sierksma, G.: Linear and integer programming: theory and practice. CRC (2002)
15. von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17, 395–416 (2007)
16. Wang, J., Do, H., Woznica, A., Kalousis, A.: Metric learning with multiple kernels. In: Advances in Neural Information Processing Systems. MIT Press (2011)
17. Weinberger, K., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbor classification. In: Advances in Neural Information Processing Systems, vol. 18, MIT Press (2006)
18. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research 10, 207–244 (2009)
19. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning with application to clustering with side-information. In: Advances in Neural Information Processing Systems. MIT Press (2003)
20. Yang, Z., Laaksonen, J.: Regularized Neighborhood Component Analysis. In: Ersbøll, B.K., Pedersen, K.S. (eds.) SCIA 2007. LNCS, vol. 4522, pp. 253–262. Springer, Heidelberg (2007)

Massively Parallel Feature Selection: An Approach Based on Variance Preservation

Zheng Zhao, James Cox, David Duling, and Warren Sarle

SAS Institute Inc. 600 Research Drive, Cary, NC 27513, USA

Abstract. Advances in computer technologies have enabled corporations to accumulate data at an unprecedented speed. Large-scale business data might contain billions of observations and thousands of features, which easily brings their scale to the level of terabytes. Most traditional feature selection algorithms are designed for a centralized computing architecture. Their usability significantly deteriorates when data size exceeds hundreds of gigabytes. High-performance distributed computing frameworks and protocols, such as the Message Passing Interface (MPI) and MapReduce, have been proposed to facilitate software development on grid infrastructures, enabling analysts to process large-scale problems efficiently. This paper presents a novel large-scale feature selection algorithm that is based on variance analysis. The algorithm selects features by evaluating their abilities to explain data variance. It supports both supervised and unsupervised feature selection and can be readily implemented in most distributed computing environments. The algorithm was developed as a SAS High-Performance Analytics procedure, which can read data in distributed form and perform parallel feature selection in both symmetric multiprocessing mode and massively parallel processing mode. Experimental results demonstrated the superior performance of the proposed method for large scale feature selection.

Keywords: Feature selection, parallel processing, big-data.

1 Introduction

Feature selection is an effective technique for dimensionality reduction and relevance detection [1]. It improves the performance of learning models in terms of their accuracy, efficiency, and model interpretability [2]. As an indispensable component for successful data mining applications, feature selection has been used in a variety of fields, including text mining, image processing, and genetic analysis, to name a few. Continual advances in computer-based technologies have enabled corporations and organizations to collect data at an increasingly fast pace. Business and scientific data from many fields, such as finance, genomics, and physics, are often measured in terabytes (10^{12} bytes). The enormous proliferation of large-scale data sets brings new challenges to data mining techniques and requires novel approaches to address the big-data problem [3] in feature selection. Scalability is critical for large-scale data mining. Unfortunately, most

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 237–252, 2012.

existing feature selection algorithms do not scale well, and their efficiency significantly deteriorates or even becomes inapplicable, when the data size reaches hundreds of gigabytes (10^9 bytes). Efficient distributed programming protocols and frameworks, such as the Message Passing Interface (MPI) [4] and MapReduce [5], are proposed to facilitate programming on high-performance distributed computing infrastructures to handle very large-scale problems.

This paper presents a novel distributed parallel algorithm for handling large-scale problems in feature selection. The algorithm can select a subset of features that best explain (preserve) the variance contained in the data. According to how data variance is defined, the algorithm can perform either unsupervised or supervised feature selection. And for the supervised case, the algorithm also supports both regression and classification. Redundant features increase data dimensionality unnecessarily and worsen the learning performance [6, 7]. The proposed algorithm selects features by evaluating feature subsets and can therefore handle redundant features effectively. For parallel feature selection, the computation of the proposed algorithm is fully optimized and parallelized based on data partitioning. The algorithm is implemented as a SAS High-Performance Analytics procedure[1], which can read data in a distributed form and perform parallel feature selection in both symmetric multiprocessing (SMP) mode via multithreading and massively parallel processing (MPP) mode via MPI.

A few approaches have been proposed for parallel feature selection. In [8, 9, 10, 11], parallel processing is used to speed up feature selection by evaluating multiple features or feature subsets simultaneously. Since all these algorithms require each parallel processing unit to access the whole data, they do not scale well when the sample size is huge. To handle large scale problems, an algorithm needs to rely on data partitioning to ensure its scalability [12]. In [13], a parallel feature selection algorithm is proposed for logistic regression. The algorithm is implemented under the MapReduce framework and can evaluate features using a criterion obtained by approximating the objective function of the logistic regression model. After selecting each new feature, the algorithm needs to retrain its model, which is an iterative process. In contrast, the proposed algorithm solves a problem with a closed form solution in each step and therefore might be more efficient. To the best knowledge of the authors, all existing parallel feature selection algorithms are for supervised learning, while the proposed algorithm supports both supervised and unsupervised feature selection.

The contributions of this paper are: (1) The proposed algorithm provides a unified approach for both unsupervised and supervised feature selection. For supervised feature selection, it also supports both regression and classification. (2) The proposed algorithm can effectively handle redundant features in feature selection. (3) The algorithm is fully optimized and parallelized based on data partitioning, which ensures its scalability for handling large-scale problems. To the best knowledge of the authors, this is the first distributed parallel algorithm for unsupervised feature selection.

[1] A SAS procedure is a c-based routine for statistical analysis in the SAS system.

2 Maximum Variance Preservation for Feature Selection

This section presents a multivariate formulation for feature selection based on maximum variance preservation. It first shows how to use the formulation to perform unsupervised feature selection, then extends it to support supervised feature selection for both regression and classification.

2.1 Unsupervised Feature Selection

When label information is unavailable, feature selection becomes challenging. To address this issue, researchers propose various criteria for unsupervised feature selection. For example, in [14], the performance of a clustering algorithm is used to evaluate the utility of a feature subset; in [15, 16], each feature's ability to preserve locality is evaluated and used to select features; and in [17] an entropy-based criterion is proposed and used for feature selection. This paper proposes a multivariate formulation for feature evaluation in a distributed computing environment. The criterion is based on maximum variance preservation, which promotes the selection of the features that can best preserve data variance.

Assume that k features need to be selected. Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ be a data set that contains n samples, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and m features, $\mathbf{f}_1, \ldots, \mathbf{f}_m$. In this work, it is assumed that all features have been centralized to have zero mean, $\mathbf{1}^\top \mathbf{f} = \mathbf{0}$, where $\mathbf{1}$ is a column vector with all its elements being 1. Let $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$, where $\mathbf{X}_1 \in \mathbb{R}^{n \times k}$ contains the k selected features and $\mathbf{X}_2 \in \mathbb{R}^{n \times (m-k)}$ contains the remaining ones. The proposed maximum variance preservation criterion selects features by minimizing the following expression:

$$arg \min_{\mathbf{X}_1} Trace \left(\mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right) \mathbf{X}_2 \right) \tag{1}$$

Let $\mathbf{X}_1 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the singular value decomposition of $\mathbf{X}_1$, and let $\mathbf{U} = (\mathbf{U}_R, \mathbf{U}_N)$, where $\mathbf{U}_R$ contains the left singular vectors that correspond to the nonzero singular values and $\mathbf{U}_N$ contains the left singular vectors that correspond to the zero singular values. It can be verified that $\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top = \mathbf{U}_N \mathbf{U}_N{}^\top$, therefore the following equation holds:

$$Trace \left(\mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right) \mathbf{X}_2 \right) = Trace \left(\left(\mathbf{U}_N^\top \mathbf{X}_2 \right)^\top \left(\mathbf{U}_N^\top \mathbf{X}_2 \right) \right) \tag{2}$$

The columns of $\mathbf{U}_N$ span the null space of $\mathbf{X}_1^T$. Since each row of $\mathbf{X}_1^\top$ corresponds to a feature in $\mathbf{X}_1$, $\mathbf{U}_N^\top \mathbf{X}_2$ effectively projects the features in $\mathbf{X}_2$ to the null space of the features in $\mathbf{X}_1$. Therefore, Expression (1) measures the variance that resides in the null space of $\mathbf{X}_1^\top$, which is the variance that cannot be explained by the features in $\mathbf{X}_1$. And minimizing it leads to the selection of the features that can jointly explain the maximum amount of the data variance.

2.2 Supervised Feature Selection

When label information is available, Expression (1) can be extended to support feature selection in both regression and classification (categorization) settings.

The Regression Case. In a regression setting, all responses are numerical. Let $\mathbf{Y} \in \mathbb{R}^{n \times t}$ be the response matrix that contains t response vectors, and $\mathbf{X}_1$ and $\mathbf{X}_2$ are defined as before. Assume that k features need to be selected. In a regression setting, feature selection can be achieved by minimizing:

$$arg \min_{\mathbf{X}_1} Trace \left(\mathbf{Y}^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right) \mathbf{Y} \right) \tag{3}$$

where $\left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right)^{\frac{1}{2}} = \mathbf{U}_N$ projects $\mathbf{Y}$ to the null space of $\mathbf{X}_1^\top$. Expression (3) measures the response variance that resides in the null space of $\mathbf{X}_1^\top$, which is the variance of $\mathbf{Y}$ that cannot be explained by the features in $\mathbf{X}_1$. Clearly, minimizing the expression leads to selecting the features that can jointly explain the maximum amount of response variance.

The Classification Case. In a classification setting, one categorical response is specified. Let the response vector be $\mathbf{y}$ with C different values, $\{1, \ldots, C\}$. A response matrix $\mathbf{Y} \in \mathbb{R}^{n \times C}$ can be created using the following equation:

$$\mathbf{Y}_{i,j} = \begin{cases} \left(\sqrt{\frac{1}{n_j}} - \frac{\sqrt{n_j}}{n} \right), & y_i = j \\ -\frac{\sqrt{n_j}}{n}, & y_i \neq j \end{cases} \tag{4}$$

where n_j is the number of instances in class j, and $y_i = j$ denotes that the ith instance belongs to the jth class. This $\mathbf{Y}$ is first used in [18] for least square linear discriminant analysis. Applying it in Expression (3) enables feature selection in a classification setting, which leads to selecting the features that maximize the discriminant criterion of linear discriminant analysis (LDA).

Theorem 1. *Assume that features have been centralized to have zero mean and that the response matrix* $\mathbf{Y}$ *is defined by Equation (4). Minimizing Expression (3) is equivalent to maximizing the discriminant criterion of LDA,*

$$\max Trace \left(\mathbf{S}_t^{-1} \mathbf{S}_b \right) \tag{5}$$

where $\mathbf{S}_t$ *and* $\mathbf{S}_b$ *are the total and the between-class scatter matrices on* $\mathbf{X}_1$*.*

Proof. Let $\mathbf{Y}$ be defined in Equation (4), and all features have zero mean. The theorem can be proved by verifying the following equations:

$$\frac{1}{n} \mathbf{X}^\top \mathbf{X} = \mathbf{S}_t = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}_i - \mathbf{c} \right) \left(\mathbf{x}_i - \mathbf{c} \right)^\top \tag{6}$$

$$\mathbf{X}^\top \mathbf{Y} \mathbf{Y}^\top \mathbf{X} = \mathbf{S}_b = \frac{1}{n} \sum_{j=1}^{C} n_j \left(\mathbf{c}_j - \mathbf{c} \right) \left(\mathbf{c}_j - \mathbf{c} \right)^\top \tag{7}$$

In the preceding equations, $\mathbf{c}$ is the mean of the whole data. Since features have been centralized to have zero mean, in the preceding equations $\mathbf{c} = 0$. $\mathbf{x}_i$ is the ith instance, and $\mathbf{c}_j$ is the mean of the instances that belong to class j.

The discriminant criterion of LDA measures the separability of the instances from different classes. For example, Expression (5) achieves a large value when instances from the same class are close, while instances from different classes are far away from each other. When Equation (4) is applied in Expression (3) for feature selection, the features that maximize the separability of the instances from different classes are selected. This is a desirable property for classifiers.

3 The Computation

Given m features, finding the k features minimizing Expressions (1) and (3) is a combinatorial optimization problem, which is NP-hard (nondeterministic polynomial-time hard). The sequential forward selection (SFS) strategy[2] is an efficient way of generating a suboptimal solution for the problem [1]. This section derives closed form solutions for the problem based on sequential forward selection, which significantly improves its efficiency. It also presents algorithms for computing the solutions in a distributed parallel computing environment.

3.1 Closed Form Solutions Based on SFS

Solution for Unsupervised Feature Selection. Assume that q features have been selected. Let $\mathbf{X}_1$ contain the q selected features, and let $\mathbf{X}_2$ contain the remaining ones. In the $q+1$ step, a feature $\mathbf{f}$ is selected by

$$arg\min_{\mathbf{f}} Trace\left(\hat{\mathbf{X}}_2^\top\left(\mathbf{I}-\hat{\mathbf{X}}_1\left(\hat{\mathbf{X}}_1^\top\hat{\mathbf{X}}_1\right)^{-1}\hat{\mathbf{X}}_1^\top\right)\hat{\mathbf{X}}_2\right) \tag{8}$$

where $\hat{\mathbf{X}}_1$ contains $\mathbf{f}$ and the q selected features, and $\hat{\mathbf{X}}_2$ contains the remaining ones. Let $\mathbf{U}_N = \left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)^{\frac{1}{2}}$, the following theorem applies:

Theorem 2. *Solving the problem specified in Expression (8) is equivalent to maximizing the following expression:*

$$arg\max_{\mathbf{f}} \frac{\left\|\mathbf{X}_2^\top\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)\mathbf{f}\right\|_2^2}{\left\|\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)^{\frac{1}{2}}\mathbf{f}\right\|_2^2} \tag{9}$$

Proof. The theorem can be proved by applying block matrix inversion on $\hat{\mathbf{X}}_1^\top\hat{\mathbf{X}}_1$.

$$\left(\hat{\mathbf{X}}_1^\top\hat{\mathbf{X}}_1\right)^{-1} = \begin{pmatrix} \mathbf{A}^{-1}+\frac{1}{w}\mathbf{A}^{-1}\mathbf{b}\mathbf{b}^\top\mathbf{A}^{-1} & -\frac{1}{w}\mathbf{A}^{-1}\mathbf{b} \\ -\frac{1}{w}\mathbf{b}^\top\mathbf{A}^{-1} & \frac{1}{w} \end{pmatrix} \tag{10}$$

where $\mathbf{A}=\mathbf{X}_1^\top\mathbf{X}_1$, $\mathbf{b}=\mathbf{X}_1^\top\mathbf{f}$, $c=\mathbf{f}^\top\mathbf{f}$, and $w=c-\mathbf{b}^\top\mathbf{A}^{-1}\mathbf{b}$. The details of the proof is omitted due to space limit.

[2] To select k features, the sequential forward selection (SFS) strategy applies k steps of greedy search and selects one feature in each step.

Assuming that all features have zero mean, $\left\|\mathbf{X}_2^\top\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)\mathbf{f}\right\|_2^2$ in Equation (9) is the summation of the squares of the covariance between the feature $\mathbf{f}$ and all the unselected features (columns of $\mathbf{X}_2$) in the null space of $\mathbf{X}_1^\top$. And $\left\|\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)^{\frac{1}{2}}\mathbf{f}\right\|_2^2$ is the square of the variance of the feature $\mathbf{f}$ in the null space of $\mathbf{X}_1^\top$, which is used for normalization. Essentially, Expression (9) measures how well the feature $\mathbf{f}$ can explain the variance that cannot be explained by the q selected features. Compared to Expression (8), Expression (9) singles out the computations that are common for evaluating different features. This makes it possible to compute them only once in each step and therefore improves the efficiency for solving the problem.

Let m be the number of all features, n the number of samples, and k the number of features to be selected. Also assume that $m \gg k$. It is easy to verify that in a traditional centralized computing environment, the time complexity for selecting k features by solving Expression (9) is:

$$O\left(m^2\left(n+k^2\right)\right) \tag{11}$$

In the preceding expression, m^2n corresponds to the complexity for computing the covariance matrix. And m^2k^2 corresponds to selecting k features out of m.

Solution for Supervised Feature Selection. The following theorem enables efficient feature selection with Expression (3):

Theorem 3. *When the problem specified in Expression (3) is solved by sequential forward selection, in each step the selected feature* $\mathbf{f}$ *must maximize:*

$$arg\max_{\mathbf{f}} \frac{\left\|\mathbf{Y}^\top\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)\mathbf{f}\right\|_2^2}{\left\|\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)^{\frac{1}{2}}\mathbf{f}\right\|_2^2} \tag{12}$$

Proof. It can be proved in the same way as Theorem 2.

Let C be the number of columns in $\mathbf{Y}$. The time complexity of selecting k features using Expression (12) is

$$O\left(mk\left(n+k^2\right)\right) \tag{13}$$

To obtain Expression (13), it is assumed that $m \gg k > C$.

3.2 Parallel Computation through MPP and SMP

The operations for computing Expression (9) and (12) need to be carefully ordered, optimized, and parallelized to ensure efficiency and scalability.

Massive Parallel Processing (MPP). The master-worker/slave architecture based on MPI [4] is used to support massive parallel processing. In this architecture, given $p+1$ parallel processing units, one unit is used as the master for control, and the remaining p units is used as workers for computation. In the implementation, all expensive operations for computing feature relevance are properly decomposed, so that they can be computed in parallel based on data partitioning. Assume that a data set has n instances and m features. p homogenous computers (the workers) are available. A data partitioning technique evenly distributes instances to the workers, so that each worker obtain $\frac{n}{p}$ instances for computation. It is shown in [20] that any operation fitting the Statistical Query model[3] can be computed in parallel based on data partitioning. Studies also showed that when data size is large enough, parallelization based on data partitioning can result in linear speedup as computing resources increase [20, 12]. Algorithms 1 and 2 contain the implementation details for distributed parallel feature selection based on MPI. The validness of the computation can be verified by decomposing operations into various summation forms over instances. The details about the verification is not presented due to space limit.

Symmetric Multiprocessing (SMP). Solving the problems specified in Expression (9) and (12) involves a series of matrix-vector operations. These operations are packed together and rewritten in the matrix-matrix operation form. This effectively simplifies programming and allows developers to use a highly optimized threaded BLAS library to speed up computation on the workers through multi-threading. As an example, in unsupervised feature selection, let $t_{i_{r,j}} = \mathbf{f}_{i_{r,j}}^\top \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top \mathbf{f}_{i_{r,j}}$, where $\mathbf{f}_{i_{r,j}}$ is the j-th feature on the r-th worker. $(t_{i_{r,1}}, \ldots, t_{i_{r,\frac{m}{p}}})$ can be computed as, $(t_{i_{r,1}}, \ldots, t_{i_{r,\frac{m}{p}}}) = \mathbf{1}^\top (\mathbf{B}_r \otimes \mathbf{E}_r)$, where $\otimes$ denotes element-wise matrix multiplication. Let $\mathbf{X}_r = (\mathbf{f}_{i_{r,1}}, \ldots, \mathbf{f}_{i_{r,\frac{m}{p}}})$ and $\mathbf{A} = \mathbf{X}_1^\top \mathbf{X}_1$, it can be verified that $\mathbf{B}_r = \mathbf{X}_1^\top \mathbf{X}_r$, and $\mathbf{E}_r = \mathbf{A}^{-1}\mathbf{B}_r$.

3.3 The Implementations

Algorithm 1 and 2 contain the pseudocode for unsupervised and supervised feature selection respectively. Both algorithms assume that the data have been properly partitioned and distributed to p worker nodes. In the algorithms, $\otimes$ and $\oslash$ denote element-wise matrix multiplication and division, respectively.

For unsupervised feature selection, the covariance among features is used repeatedly in the evaluation process. Therefore, it is more efficient to compute the whole covariance matrix $\mathbf{C}$ before feature selection. In Algorithm 1, **Line 2** to **Line 5** compute feature scores to select the first feature. Since no feature has been selected, Expression (9) can be simplified to $\frac{\|\mathbf{X}^\top \mathbf{f}_i\|_2^2}{\mathbf{f}_i^\top \mathbf{f}_i} = \frac{\|\mathbf{c}^i\|_2^2}{c_{i,i}}$, where $\mathbf{c}^i$ is the ith column of $\mathbf{C}$, and $C_{i,i}$ is the ith diagonal element. In **Line 2**, $\mathbf{v}_r$ contains the

[3] An operation fits the Statistical Query model if it can be decomposed and written in summation forms over the instances.

Input: $\mathbf{X}_1, \ldots, \mathbf{X}_p, \in \mathbb{R}^{\frac{n}{p} \times m}$; k
Output: $\mathbb{L}$, a list of k selected features

1 Compute covariance matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$, and distribute the rth section of the covariance matrix, $\mathbf{C}_r \in \mathbb{R}^{m \times \frac{m}{p}}$, on the rth worker, $r = 1, \ldots, p$;

2 Compute local feature scores on each **worker**

$$\mathbf{s}_r = \mathbf{1}^\top \left(\mathbf{C}_r \bigotimes \mathbf{C}_r\right), \ \mathbf{s}_r = \mathbf{s}_r \bigoslash \mathbf{v}_r; \ \mathbf{v}_r = \left(C_{i_{r,1},i_{r,1}}, \ldots, C_{i_{r,\frac{m}{p}},i_{r,\frac{m}{p}}}\right) \quad (14)$$

3 **Workers** send $\mathbf{s}_r$ to the master via MPI_Gather;

4 On the **master**, select $i = arg\max\left(s_i \mid s_i \in (\mathbf{s}_1, \ldots, \mathbf{s}_p)\right)$;

5 Initialization, $\mathbb{L} = \{F_i\}$, $l = 1$;

6 **while** $l < k$ **do**

7 The **master** sends $\mathbb{L}$ to all workers via MPI_Bcast;

8 The **worker** that contains $\mathbf{c}^i$, the ith column of $\mathbf{C}$, sends $\mathbf{c}^i$ to all other workers via MPI_Bcast;

/* ------------------simultaneously------------------- */

9 **Workers** construct $\mathbf{A}^{-1} \in \mathbb{R}^{l \times l}$, $\mathbf{B}_r \in \mathbb{R}^{l \times t_r}$, $\mathbf{D}_r \in \mathbb{R}^{(m-l) \times t_r}$, $\mathbf{v}_r \in \mathbb{R}^{t_r \times 1}$, $\mathbf{C}_{2,1} \in \mathbb{R}^{(m-l) \times l}$;

10 **Workers** compute local feature scores

$$\mathbf{E}_r = \mathbf{A}^{-1}\mathbf{B}_r, \quad \mathbf{H}_r = \mathbf{C}_{2,1}\mathbf{E}_r, \quad \mathbf{G}_r = \mathbf{D}_r - \mathbf{H}_r, \quad (15)$$

$$\mathbf{g}_r = \mathbf{1}^\top \left(\mathbf{G}_r \bigotimes \mathbf{G}_r\right), \mathbf{w}_r = \mathbf{v}_r - \mathbf{1}^\top \left(\mathbf{B}_r \bigotimes \mathbf{E}_r\right), \mathbf{s}_r = \mathbf{g}_r \bigoslash \mathbf{w}_r \quad (16)$$

Workers send $\mathbf{s}_r$ to the master via MPI_Gather;

/* --- */

11 **Master** selects $i = arg\max\left(s_i \mid s_i \in (\mathbf{s}_1, \ldots, \mathbf{s}_p)\right)$, $\mathbb{L} = \mathbb{L} \cup \{F_i\}$, $l++$;

12 **end**

Algorithm 1. Distributed parallel unsupervised feature selection.

diagonal elements of $\mathbf{C}$ that corresponds to the variance of the features on the rth worker. The vector $\mathbf{s}_r$ contains the scores of the features on the rth worker. After a feature F_i has been selected, each worker updates $\mathbf{A}^{-1}$, $\mathbf{B}_r$, $\mathbf{D}_r$, $\mathbf{v}_r$, and $\mathbf{C}_{2,1}$ in **Line 9** using $\mathbf{C}_r$ and $\mathbf{c}^i$. Let $\mathbb{L}$ contain the index of selected features, $\mathbb{L}_r$ contain the index of unselected features on the rth worker, and $\mathbb{L}_u$ contain the index of all unselected features. $\mathbf{A} = \mathbf{X}_1^\top \mathbf{X}_1 = \mathbf{C}_{\mathbb{L}\times\mathbb{L}}$, $\mathbf{B}_r = \mathbf{X}_1^\top \mathbf{X}_r = \mathbf{C}_{\mathbb{L}\times\mathbb{L}_r}$, $\mathbf{D}_r = \mathbf{X}_2^\top \mathbf{X}_r = \mathbf{C}_{\mathbb{L}_u\times\mathbb{L}_r}$, $\mathbf{C}_{2,1} = \mathbf{X}_2^\top \mathbf{X}_1$, and $\mathbf{v}_r$ contains the variance of the unselected features on the rth worker. The scores of the features on the rth worker is computed in **Line 10**. Assume that the $\mathbf{A}^{-1}$ in **Line 9** can be computed by applying rank-one update, and a tree-based mechanism is used to implement `MPI_Bcast` and `MPI_Reduce`. The total time complexity of Algorithm 1 is

$$CPU\left(\frac{m^2\left(n+k^2\right)}{p} + m^2 \log p\right) + NET\left(m^2 \log p\right) \quad (17)$$

In the preceding expressions, $CPU\left(\cdot\right)$ and $NET\left(\cdot\right)$ denote the time used for computation and for network communication, respectively.

Input: $\mathbf{X}_1, \ldots, \mathbf{X}_p \in \mathbb{R}^{\frac{n}{p} \times m}$, $\mathbf{Y}_1, \ldots, \mathbf{Y}_p \in \mathbb{R}^{\frac{n}{p} \times C}$, k
Output: $\mathbb{L}$, a list of k selected features

1 On each **worker**, compute $\mathbf{E}_r \in \mathbb{R}^{C \times m}$, $\mathbf{v}_r \in \mathbb{R}^{1 \times m}$:

$$\mathbf{E}_r = \mathbf{Y}_r^\top \mathbf{X}_r, \quad \mathbf{v}_r = \mathbf{1}^\top \left(\mathbf{X}_r \bigotimes \mathbf{X}_r\right); \tag{18}$$

2 Send $\mathbf{E}_r$ and $\mathbf{v}_r$ to the master via MPI_Reduce with MPI_SUM option:

$$\mathbf{E} = \sum_{r=1}^{p} \mathbf{E}_r, \quad \mathbf{v} = \sum_{r=1}^{p} \mathbf{v}_r; \tag{19}$$

3 On the **master**, compute feature scores

$$\mathbf{s} = \mathbf{1}^\top \left(\mathbf{E} \bigotimes \mathbf{E}\right), \quad \mathbf{s} = \mathbf{s} \bigoslash \mathbf{v}; \tag{20}$$

4 On the **master,** select $i = arg\max\left(s_i \mid s_i \in \mathbf{s}\right)$;
5 Initialization, $\mathbb{L} = \{F_i\}$, $l = 1$;
6 **while** $l < k$ **do**
7 The **master** sends $\mathbb{L}$ to all workers via MPI_Bcast;
/* ----------simultaneously---------- */
8 **Workers** compute $\mathbf{c}_r^i = \mathbf{X}_r^\top \mathbf{f}_r^i$, $\mathbf{c}_r^i \in \mathbb{R}^{m \times 1}$;
9 **Workers** send $\mathbf{c}_r^i$ to the master via MPI_Reduce with MPI_SUM option

$$\mathbf{c}^i = \sum_{r=1}^{p} \mathbf{c}_r^i, \ \mathbf{c}^i \in \mathbb{R}^{m \times 1} \tag{21}$$

/* ---------- */
10 On the **master**, construct $\mathbf{A}^{-1} \in \mathbb{R}^{l \times l}$, $\mathbf{C}_{\mathbf{Y},1} \in \mathbb{R}^{C \times l}$, $\mathbf{C}_{1,2} \in \mathbb{R}^{l \times (m-l)}$, $\mathbf{C}_{\mathbf{Y},2} \in \mathbb{R}^{C \times (m-l)}$, $\mathbf{v}_2 \in \mathbb{R}^{1 \times (m-l)}$;
11 On the **master**, compute

$$\mathbf{B} = \mathbf{A}^{-1}\mathbf{C}_{1,2}, \quad \mathbf{H} = \mathbf{C}_{\mathbf{Y},1}\mathbf{B}, \quad \mathbf{G} = \mathbf{C}_{\mathbf{Y},2} - \mathbf{H}; \tag{22}$$

$$\mathbf{g} = \mathbf{1}^\top \left(\mathbf{G} \bigotimes \mathbf{G}\right), \quad \mathbf{w} = \mathbf{v}_2 - \mathbf{1}^\top \left(\mathbf{C}_{1,2} \bigotimes \mathbf{B}\right), \quad \mathbf{s} = \mathbf{g} \bigoslash \mathbf{w}; \tag{23}$$

12 **Master** selects $i = arg\max\left(s_i \mid s_i \in \mathbf{s}\right)$, $\mathbb{L} = \mathbb{L} \cup \{F_i\}$, $l++$;
13 **end**

Algorithm 2. Distributed parallel supervised feature selection.

For supervised feature selection, only a small portion of the covariance matrix is needed for feature evaluation. Therefore, the covariance matrix is not computed before feature selection. In Algorithm 2, **Line 1** to **Line 3** compute feature scores to select the first feature. Since no feature has been selected, Expression (12) simplifies to $\frac{\|\mathbf{X}^\top \mathbf{f}\|_2^2}{\mathbf{f}^\top \mathbf{f}}$. In **Line 10**, $\mathbf{A} = \mathbf{X}_1^\top \mathbf{X}_1$, $\mathbf{C}_{\mathbf{Y},1} = \mathbf{Y}^\top \mathbf{X}_1$, $\mathbf{C}_{\mathbf{Y},2} = \mathbf{Y}^\top \mathbf{X}_2$, $\mathbf{C}_{1,2} = \mathbf{X}_1^\top \mathbf{X}_2$, and $\mathbf{v}_2$ contains the variance of the unselected features. As both $\mathbf{A}^{-1}$ and $\mathbf{B}$ can be obtained by incrementally updating their previous versions, the complexity for selecting k features using Algorithm 2 is

$$CPU\left(mk\left(\frac{n}{p}+k^2\right)\right)+NET\left(m\left(C+k\right)\log p\right) \tag{24}$$

In the preceding expression, C is the number of columns in $\mathbf{Y}$.

Expression (17) and (24) suggest that when the number of instances is large and the network is fast enough, Algorithms 1 and 2 can speed up feature selection linearly as the number of parallel processing units increases.

4 Connections to Existing Methods

In an unsupervised setting, principal component analysis (PCA) [19] also reduces dimensionality by preserving data variance. The key difference between PCA and the proposed method is that PCA generates a small set of new features (feature extraction) by linearly combining the original features, while the proposed method selects a small set of the original features (feature selection). The features returned by the proposed method are the original ones. This is very important in applications where retaining the original features is useful for model exploration or interpretation (for example, genetic analysis and text mining).

In a regression setting, let $\mathbf{f}$ be a feature vector, it can be shown that

$$\mathbf{f}^\top\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)\mathbf{Y}=\mathbf{f}^\top\left(\mathbf{Y}-\mathbf{X}_1\mathbf{W}_1\right) \tag{25}$$

where $\mathbf{W}_1=\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\mathbf{Y}$ is the solution of a least squares regression. Let $\mathbf{R}$ be the residual, $\mathbf{R}=\mathbf{Y}-\mathbf{X}_1\mathbf{W}_1$. Expression (12) can be simplified to:

$$arg\max_{\mathbf{f}}\frac{\left\|\mathbf{f}^\top\mathbf{R}\right\|_2^2}{\left\|\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)^{\frac{1}{2}}\mathbf{f}\right\|_2^2} \tag{26}$$

Therefore, in each step the proposed method selects the feature that has the largest normalized correlation with the current residual. This shows that in a regression setting the proposed method forms a special type of stepwise regression with Expression (12) as the selection criterion.

When used in a classification setting, the proposed method selects features with the discriminant criterion of LDA. LDA also reduces dimensionality. As for PCA, the key difference is that LDA generates a small set of new features, while the proposed method selects a small set of the original features.

5 Experimental Study

The proposed method was implemented as the HPREDUCE procedure based on SAS High-Performance Analytics foundation. This section evaluates its performance for both supervised and unsupervised feature selection. In the experiment,

12 representative feature selection algorithms are used for comparison. For unsupervised feature selection, six algorithms are selected as baselines: Laplacian score [15], SPEC-1 and SPEC-3 [16], trace-ratio [21], HSIC [22], and SPFS [23]. For supervised feature selection, in the classification setting, seven algorithms are compared: ReliefF [24], Fisher Score [25], trace-ratio, HSIC, mRMR [7], AROM-SVM [26], and SPFS. In the regression setting, LARS [27], and LASSO [28] are compared. Among the 12 algorithms, AROM-SVM, mRMR, SPFS, LARS and LASSO can handle redundant features.

Table 1. Summary of the benchmark data sets

Data Set	Features	Instances	Classes	Data Set	Features	Instances	Classes
RELATH	4,322	1,427	2	ORL	10,000	100	10
PCMAC	3,289	1,943	2	CRIME	147	2,215	-
AR	2,400	130	10	SLICELOC	386	53,500	-
PIE	2,400	210	10	s25mf5k	5,000	25,000,000	-
PIX	10,000	100	10	u10mf5k	5,000	10,000,000	-

Ten benchmark data sets are used in the experiment. Four are face image data: AR[4], PIE[5], PIX[6], and ORL[7] (images from 10 persons are used). Two are text data extracted from the 20-newsgroups data[8]: RELATH (BASEBALL vs. HOCKEY) and PCMAC (PC vs. MAC). Two are UCI data: CRIME (Communities and Crime Unnormalized) and SLICELOC (relative location of CT slices on axial axis)[9]. And two are large-scale data sets for performance tests. The u10mf5k data set contains 5,000 features and 10 million instances, which is used for testing unsupervised feature selection. The s25mf5k data set contains 5,000 features, 1 response, and 25 million instances, which is used for testing supervised feature selection. Each data set has 100 continuous variables sampled from uniform distribution. And the remains are binary variables sampled from Bernoulli distribution. Details on the ten data sets can be found in Table 1. The first six data sets are used to test unsupervised feature selection and supervised feature selection for classification. The seventh and the eighth data sets are used to test feature selection for regression. And the last two are used to evaluate the HPREDUCE procedure in a distributed computing environment.

Assume that $\mathbb{L}$ is the set of selected features and that $\mathbf{X}_{\mathbb{L}}$ is the data that contain only features in $\mathbb{L}$. For the classification setting, algorithms are compared on (1) classification accuracy and (2) redundancy rate which is defined as:

$$RED\left(\mathbb{L}\right) = \frac{1}{m(m-1)} \sum_{F_i, F_j \in \mathbb{L}, i>j} \rho_{i,j} \tag{27}$$

[4] http://rvl1.ecn.purdue.edu/~leix/aleix_face_DB.html
[5] http://peipa.essex.ac.uk/ipa/pix/faces/manchester/
[6] http://www.ri.cmu.edu/projects/project_418.html
[7] http://www.uk.research.att.com/facedatabase.html
[8] http://people.csail.mit.edu/jrennie/20Newsgroups/
[9] http://archive.ics.uci.edu/ml/index.html

where $\rho_{i,j}$ returns the correlation between feature F_i and feature F_j. Equation (27) assesses the average correlation among all feature pairs. A large value indicates that features in $\mathbb{L}$ are strongly correlated and thus redundant features might exist. In the regression setting, algorithms are compared on (1) rooted mean square error (RMSE) and (2) redundancy rate. For unsupervised feature selection, algorithms are compared on: (1) redundancy rate and (2) percentage of the total variance explained by features in $\mathbb{L}$,

$$PCT_{VAR}(\mathbb{L}) = \frac{Trace\left(\mathbf{X}^\top \mathbf{X}_\mathbb{L} \left(\mathbf{X}_\mathbb{L}^\top \mathbf{X}_\mathbb{L}\right)^{-1} \mathbf{X}_\mathbb{L}^\top \mathbf{X}\right)}{Trace\left(\mathbf{X}^\top \mathbf{X}\right)} \tag{28}$$

For each data set, half of the instances are randomly sampled for training and the remaining are used for test. The process is repeated 20 times, which results in 20 different partitions of the data set. Each feature selection algorithm is used to select $5, 10, \ldots, 100$ features on each partition. The obtained 20 feature subsets are then evaluated using a criterion $\mathcal{C}$. By doing this, a score matrix $\mathbf{S} \in \mathbb{R}^{20\times 20}$ is generated for each algorithm, where each row of $\mathbf{S}$ corresponds to a data partition and each column corresponds to a size of the feature subset. The average score of $\mathcal{C}$ is obtained by $s = \frac{\mathbf{1}^\top \mathbf{S} \mathbf{1}}{20\times 20}$. To calculate classification accuracy, linear support vector machine (SVM) is used. The parameters of SVM and all feature selection algorithms are tuned via 5 fold cross-validation on the training data. Let $\mathbf{s} = \frac{\mathbf{1}^\top \mathbf{S}}{20}$. The elements of $\mathbf{s}$ corresponds to the average score achieved when different numbers of features are selected. The paired Student's t test is applied to compare the $\mathbf{s}$ achieved by different algorithms to $\mathbf{s}^*$, the best $\mathbf{s}$ measured by $\mathbf{1}^\top \mathbf{s}$. And the threshold for rejecting the null hypothesis is set to 0.05. Rejecting the null hypothesis means that $\mathbf{s}$ and $\mathbf{s}^*$ are significantly different, and suggests that the performance of the algorithm is consistently different to the best algorithm when different numbers of selected features.

5.1 Study of Unsupervised Cases

Percentage of Explained Variance: Table 2 presents the average percentage of the data variance explained by the features selected by different algorithms. The result shows that compared with the baselines, the HPREDUCE procedure achieved the best performance on all six data sets. This is to be expected, since the HPREDUCE procedure is designed to preserve data variance. The result demonstrates the strong capability of the proposed algorithm for preserving variance in feature selection. It also suggests that using Expression (9) with sequential forward search is effective for minimizing Expression (1).

Redundancy Rate: Table 3 presents the average redundancy rate results. It shows that SPFS and the HPREDUCE procedure achieved much better results than the others. This is to be expected, since they are designed to handle redundant features, while the others are not.

Table 2. Unsupervised feature selection: explained variance with p-val

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
Laplacian	0.13 (.00)	0.10 (.00)	0.57 (.00)	0.76 (.00)	0.55 (.00)	0.45 (.00)	0.427	0
SPEC-1	0.13 (.00)	0.10 (.00)	0.57 (.00)	0.75 (.00)	0.56 (.00)	0.45 (.00)	0.427	0
SPEC-3	0.21 (.00)	0.18 (.00)	0.61 (.00)	0.78 (.00)	0.58 (.00)	0.52 (.00)	0.481	0
Trace-ratio	0.44 (.00)	0.45 (.00)	0.57 (.00)	0.75 (.00)	0.56 (.00)	0.45 (.00)	0.537	0
HSIC	0.42 (.00)	0.44 (.00)	0.62 (.00)	0.75 (.00)	0.55 (.00)	0.45 (.00)	0.538	0
SPFS	0.45 (.00)	0.47 (.01)	0.74 (.01)	0.86 (.00)	0.72 (.01)	0.60 (.01)	0.639	0
HPREDUCE	**0.60** (1.0)	**0.54** (1.0)	**0.97** (1.0)	**0.97** (1.0)	**0.96** (1.0)	**0.97** (1.0)	**0.835**	6

Table 3. Unsupervised feature selection: redundancy rate with p-val

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
Laplacian	0.70 (.00)	0.78 (.00)	0.90 (.00)	0.85 (.00)	0.82 (.00)	0.85 (.00)	0.817	0
SPEC-1	0.71 (.00)	0.78 (.00)	0.90 (.00)	0.87 (.00)	0.80 (.00)	0.85 (.00)	0.818	0
SPEC-3	0.84 (.00)	0.93 (.00)	0.89 (.00)	0.81 (.00)	0.78 (.00)	0.73 (.00)	0.829	0
Trace-ratio	0.20 (.00)	0.27 (.00)	0.90 (.00)	0.87 (.00)	0.80 (.00)	0.85 (.00)	0.649	0
HSIC	0.17 (.00)	0.25 (.00)	0.90 (.00)	0.84 (.00)	0.80 (.00)	0.85 (.00)	0.633	0
SPFS	0.08 (.00)	0.11 (.00)	0.36 (.00)	**0.31** (1.0)	**0.24** (1.0)	**0.26** (.05)	0.227	3
HPREDUCE	**0.02** (1.0)	**0.02** (1.0)	**0.22** (1.0)	0.34 (.01)	0.27 (.01)	**0.22** (1.0)	**0.181**	4

5.2 Study of Supervised Cases

Classification, Accuracy: Table 4 presents the average accuracy achieved by SVM using the features selected by algorithms. The HPREDUCE procedure achieved the best results on five data sets, which is followed by SPFS (three data sets) and Arom-SVM (two data sets). According to the average accuracy, the HPREDUCE procedure also performed the best (0.880), followed by SPFS (0.869) and HSIC (0.813). This result demonstrates the good performance of the HPREDUCE procedure in the classification setting.

Table 4. Supervised feature selection for classification: accuracy with p-val

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
ReliefF	0.70 (.00)	0.66 (.00)	0.92 (.00)	0.92 (.00)	0.76 (.00)	0.78 (.00)	0.789	0
Fisher Score	**0.86** (1.0)	0.73 (.00)	0.92 (.00)	0.90 (.01)	0.72 (.00)	0.73 (.00)	0.810	1
Trace-ratio	**0.86** (1.0)	0.73 (.00)	0.92 (.00)	0.90 (.01)	0.72 (.00)	0.73 (.00)	0.810	1
HSIC	**0.85** (.14)	0.75 (.00)	0.92 (.00)	0.90 (.01)	0.72 (.00)	0.74 (.00)	0.813	1
mRMR	0.84 (.00)	**0.79** (.81)	0.85 (.00)	0.92 (.02)	0.64 (.00)	0.68 (.00)	0.787	1
Arom-SVM	**0.85** (.14)	0.75 (.00)	0.80 (.00)	**0.90** (.09)	0.55 (.00)	0.71 (.00)	0.761	2
SPFS	**0.85** (.32)	0.78 (.02)	0.95 (.02)	**0.94** (.14)	**0.80** (.13)	0.89 (.00)	0.869	3
HPREDUCE	0.84 (.00)	**0.80** (1.0)	**0.96** (1.0)	**0.95** (1.0)	**0.81** (1.0)	**0.92** (1.0)	**0.880**	5

Classification, Redundancy Rate: The average redundancy rate achieved by algorithms are presented in Table 5. Among the eight algorithms in the table, mRMR, Arom-SVM, SPFS, and the HPREDUCE procedure are designed to handle redundant features. In the experiment, on average these algorithms achieved redundancy rates at the level of 0.2. In contrast, the other four algorithms had much higher redundancy rates. The result shows that the HPREDUCE procedure is effective in handling redundant features for classification.

Table 5. Supervised feature selection for classification: redundancy rate with p-val

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
ReliefF	0.10 (.00)	0.09 (.00)	0.78 (.00)	0.38 (.00)	0.76 (.00)	0.89 (.00)	0.501	0
Fisher Score	0.07 (.00)	0.15 (.00)	0.83 (.00)	0.40 (.00)	0.67 (.00)	0.77 (.00)	0.481	0
Trace-ratio	0.07 (.00)	0.15 (.00)	0.83 (.00)	0.40 (.00)	0.67 (.00)	0.77 (.00)	0.481	0
HSIC	0.13 (.00)	0.10 (.00)	0.83 (.00)	0.40 (.00)	0.67 (.00)	0.77 (.00)	0.483	0
mRMR	**0.04** (1.0)	0.04 (.00)	0.33 (.00)	**0.26** (.46)	**0.25** (1.0)	**0.25** (1.0)	**0.194**	4
Arom-SVM	0.05 (.00)	0.07 (.00)	**0.26** (1.0)	0.29 (.02)	**0.25** (.22)	**0.25** (.35)	0.196	3
SPFS	0.11 (.00)	0.07 (.00)	0.45 (.00)	**0.25** (1.0)	0.31 (.03)	0.36 (.00)	0.260	1
HPREDUCE	0.05 (.00)	**0.03** (1.0)	0.32 (.00)	0.31 (.00)	0.31 (.00)	0.27 (.00)	0.214	1

Regression: In the regression setting, the HPREDUCE procedure is compared to LARS and LASSO. The RMSE and redundancy rate results are presented in Tables 6, respectively. The results suggest that in terms of RMSE and redundancy rate, the performance of the three algorithms are largely comparable on the benchmark data sets. Compared to LARS and LASSO, the HPREDUCE procedure is a general method for both supervised and unsupervised feature selection, while LARS and LASSO are for supervised regression only.

Table 6. Supervised feature selection for regression, RMSE (col 2- col 4), the lower the better; redundancy rate (col 5 - col 7) with p-val

DATA	LARS	LASSO	HPREDUCE	LARS	LASSO	HPREDUCE
CRIME	3.6e-7 (.00)	3.6e-7 (.00)	**3.3e-7** (1.0)	**0.31** (1.0)	**0.31** (1.0)	0.32 (.00)
SLICELOC	2.8e-3 (.04)	2.8e-3 (.04)	**2.6e-3** (1.0)	0.17 (.00)	0.17 (.00)	**0.14** (1.0)
Average	1.38e-3	1.38e-3	**1.32e-3**	0.241	0.241	**0.233**
Best	0	0	2	1	1	1

5.3 Study of Scalability

To evaluate the scalability of the HPREDUCE procedure, it was tested in a distributed computing environment. The cluster has 32 nodes, and each node has two Intel Xeon CPUs, 16 GB memory, and two 186GB disk drives. In the experiment, different numbers of workers are used for selecting 200 features from the input data. Compared with the unsupervised case, supervised feature selection with the HPREDUCE procedure has a lower time complexity. Therefore, for supervised feature selection the maximum number of nodes is set to 20, while for unsupervised feature selection, this number is increased to 30. Multiple threads are used on each node for matrix computation.

The running time and the speedup information for both supervised and unsupervised feature selection is presented in Figure 1. It shows that the HPREDUCE procedure generally performs faster when more computing resource is available. For example, when only one worker node is used for computation in the unsupervised case, the HPREDUCE procedure finishes in 1,670.98 seconds. When 30 worker nodes are used, it finishes in just 83.69 seconds. In general, for both supervised and unsupervised feature selection, the speedup of the HPREDUCE procedure is linear. For the supervised case, the speedup ratio (slope of the line) of the HPREDUCE procedure is close to 1, which is quite good. And for the unsupervised case, the speedup ratio is about 0.66. The unsupervised case has

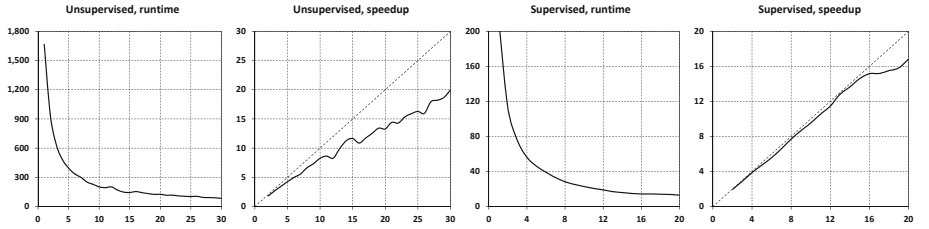


Fig. 1. Runtime and speedup in the unsupervised and the supervised settings with different number of workers for feature selection

a lower speedup ratio because it involves more network communication between the master and the workers in the feature selection process. It can also be observed from the s25mf5k data set that when more than 15 nodes are used for supervised feature selection, the speedup ratio of the HPREDUCE procedure decreases. For a fixed size problem, when too many nodes are used, the warm-up and the communication costs start to offset the increase of computing resources. The results clearly demonstrate the scalability of the proposed algorithm.

6 Conclusions

This paper presents a distributed parallel feature selection algorithm based on maximum variance preservation. The proposed algorithm forms a unified approach for feature selection. By defining the preserving target in different ways, the algorithm can achieve both supervised and unsupervised feature selection. And for supervised feature selection, it also supports both regression and classification. The algorithm performs feature selection by evaluating feature sets and can therefore handle redundant features. The computation of the algorithm is also optimized and parallelized to support both MPP an SMP. As illustrated by an extensive experimental study, the proposed algorithm can effectively remove redundant features and achieve superior performance for both supervised and unsupervised feature selection. The study also shows that given a large-scale data set, the proposed algorithm can significantly improve the efficiency of feature selection through distributed parallel computing. Our ongoing work will extend the HPREDUCE procedure to also support semi-supervised feature selection and sparse feature extraction, such as sparse PCA and sparse LDA.

Acknowledgments. The authors would like to thank An Shu, Anne Baxter, Russell Albright, and the anonymous reviewers for their valuable suggestions to improve this paper.

References

[1] Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, Boston (1998)

[2] Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182 (2003)
[3] Zaki, M.J., Ho, C.T. (eds.): Large-scale parallel data mining. Springer (2000)
[4] Snir, M., et al.: MPI: The Complete Reference. MIT Press, Cambridge (1995)
[5] Dean, J., Ghemawat, S.: System and method for efficient large-scale data processing, United States Patent 7650331 (2010)
[6] Hall, M.: Correlation-Based Feature Selection for Machine Learning. PhD thesis, University of Waikato, Dept. of Computer Science (1999)
[7] Ding, C., Peng, H.: Minimum redundancy feature selection from microarray gene expression data. In: Proceedings of the CSB, pp. 523–529 (2003)
[8] Felix, G.L., et al.: Solving feature subset selection problem by a parallel scatter search. European Journal of Operational Research 169(2), 477–489 (2006)
[9] Melab, N., et al.: Grid computing for parallel bioinspired algorithms. Journal of Parallel and Distributed Computing 66(8), 1052–1061 (2006)
[10] Garcia, D.J., et al.: A parallel feature selection algorithm from random subsets. In: Proceedings of the International Workshop on Parallel Data Mining (2006)
[11] Guillén, A., Sorjamaa, A., Miche, Y., Lendasse, A., Rojas, I.: Efficient Parallel Feature Selection for Steganography Problems. In: Cabestany, J., Sandoval, F., Prieto, A., Corchado, J.M. (eds.) IWANN 2009, Part I. LNCS, vol. 5517, pp. 1224–1231. Springer, Heidelberg (2009)
[12] Kent, P., Schabenberger, O.: SAS high performance computing: The future is not what it used to be (2011), http://www.monash.com/uploads/SAS_HPA_2011-Longer.pdf
[13] Singh, S., et al.: Parallel large scale feature selection for logistic regression. In: Proc. of SDM (2009)
[14] Dy, J.G., Brodley, C.E.: Feature selection for unsupervised learn. Journal of Machine Learning Research 5, 845–889 (2004)
[15] He, X., et al.: Laplacian score for feature selection. In: Proc. of NIPS (2005)
[16] Zhao, Z., Liu, H.: Spectral feature selection for supervised and unsupervised learning. In: Proceedings of ICML (2007)
[17] Dash, M., et al.: Feature selection for clustering, a filter solution. In: Proceedings of ICDM (2002)
[18] Ye, J.: Least squares linear discriminant analysis. In: Proceedings of ICML (2007)
[19] Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Springer (2002)
[20] Chu, C.T., et al.: Map-reduce for machine learning on multicore. In: Proceedings of NIPS (2007)
[21] Nie, F., et al.: Trace ratio criterion for feature selection. In: Proc. of AAAI (2008)
[22] Song, L., et al.: Supervised feature selection via dependence estimation. In: Proceedings of ICML (2007)
[23] Zhao, Z., Wang, L., Liu, H., Ye, J.: On similarity preserving feature selection. IEEE Transactions on Knowledge and Data Engineering 99, 198–206 (2011)
[24] Sikonja, M.R., Kononenko, I.: Theoretical and empirical analysis of Relief and ReliefF. Machine Learning 53, 23–69 (2003)
[25] Duda, R., et al.: Pattern Classification, 2nd edn. John Wiley & Sons (2001)
[26] Weston, J., et al.: Use of the zero norm with linear models and kernel methods. Journal of Machine Learning Research 3, 1439–1461 (2003)
[27] Efron, B., et al.: Least angle regression. Annals of Statistics 32, 407–449 (2004)
[28] Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B 58(1), 267–288 (1994)

PCA, Eigenvector Localization and Clustering for Side-Channel Attacks on Cryptographic Hardware Devices

Dimitrios Mavroeidis, Lejla Batina, Twan van Laarhoven, and Elena Marchiori

Institute for Computing and Information Sciences,
Radboud University Nijmegen, The Netherlands

Abstract. Spectral methods, ranging from traditional Principal Components Analysis to modern Laplacian matrix factorization, have proven to be a valuable tool for a wide range of diverse data mining applications. Commonly these methods are stated as optimization problems and employ the extremal (maximal or minimal) eigenvectors of a certain input matrix for deriving the appropriate statistical inferences. Interestingly, recent studies have questioned this "modus operandi" and revealed that useful information may also be present within low-order eigenvectors whose mass is concentrated (localized) in a small part of their indexes. An application context where localized low-order eigenvectors have been successfully employed is "Differential Power Analysis" (DPA). DPA is a well studied side-channel attack on cryptographic hardware devices (such as smart cards) that employs statistical analysis of the device's power consumption in order to retrieve the secret key of the cryptographic algorithm. In this work we propose a data mining (clustering) formulation of the DPA process and also provide a theoretical model that justifies and explains the utility of low-order eigenvectors. In our data mining formulation, we consider that the key-relevant information is modelled as a "low-signal" pattern that is embedded in a "high-noise" dataset. In this respect our results generalize beyond DPA and are applicable to analogous low-signal, hidden pattern problems. The experimental results using power trace measurements from a programmable smart card, verify our approach empirically.

1 Introduction

Spectral Clustering [17] is a popular data mining paradigm that is currently considered both as an effective practical tool for data analysis and also an active area of research. The common characteristic of Spectral Clustering methods is that they employ the spectrum (eigenvectors and eigenvalues) of certain input matrices as a central component for deriving the required data inferences. For instance, Spectral Clustering for Normalized Cut optimization [17] employs the (extremal) eigenvectors of the normalized Laplacian matrix for deriving the data clustering structure. Due to their common association with Trace optimization problems, most spectral methods employ extremal (maximal or minimal) eigenvectors for drawing the necessary inferences.

Recent studies have illustrated that low-order (not extremal) eigenvectors may also be useful for detecting interesting structures within data. More precisely, in [8,7] it was demonstrated that low-order eigenvectors that are localized (i.e. contain a large number

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 253–268, 2012.

of zero or near-zero entries), can be effectively used for detecting small, local and well connected clusters. In a global sense these clusters may be difficult to detect since they are small and do not correspond to a clustering objective optimum. Interestingly, the concept of eigenvector (or more generally eigenfunction) localization is well known in several scientific applications such as Quantum Mechanics, DNA data and astronomy (see [8,7,18] and references therein).

Low-order, localized eigenvectors have been successfully employed in the context of Differential Power Analysis (DPA) [16,5]. DPA is a side-channel attack which involves statistical analysis of a cryptographic device's power consumption. Cryptographic algorithms are nowadays typically implemented in software or hardware on (small) physical devices that interact with and are influenced by their environments. These devices provide unintended output channels, called side channels. In general, these types of information leakages may be linked either to the types of operations that the cryptographic algorithm is performing, or to the data, i.e., the keys being processed. This makes them a very powerful tool for trying to extract the secret key. Using DPA, an adversary can obtain secret keys by analyzing power consumption measurements from multiple cryptographic operations performed by a vulnerable smart card or other device.

In this work we propose a novel data mining (clustering) formulation of the DPA process and also introduce a theoretical model that illustrates the utility and semantics of low-order eigenvectors. In our data mining formulation, we consider that the key-relevant information is modelled as a "low-signal" pattern that is embedded in a "high-noise" dataset. The essential property that allows for the detection of these low-signal patterns, is that they depend on a very small number of features. The embedding of such patterns in a high-noise dataset will result in the localization of the relevant eigenvectors thus making these patterns detectable, even though they are "buried" in the middle or lower part of the matrix's spectrum. In this respect our results generalize beyond DPA and are applicable to analogous low-signal/hidden pattern problems. Consequently, we employ Inverse Participation Ration (IPR) measure that can effectively select the appropriate low-order eigenvectors even in cases where standard countermeasures, that aim in hiding the key-relevant patterns from the device's power consumption, are employed.

2 Differential Power Analysis and Clustering

2.1 Differential Power Analysis (DPA)

Small embedded devices such as smart cards and mobile phones have become omnipresent in our lives as they are used daily in financial transactions, access control, mobile payments, etc. Hence, the security of such devices relies on the security protocols that are founded on standard cryptographic algorithms and in particular on their implementations. The weaknesses of cryptographic implementations are typically explored via side-channel information, e.g. power consumption of a device, which can be monitored and statistically analyzed to retrieve cryptographic secret keys of the device.

Side-channel attacks are the main security threat for smart cards since the first academic publications by Kocher et al. [14,15]. Different sources of side-channel data, such as electromagnetic emanation [21,12], timing [14], sound, and temperature have been

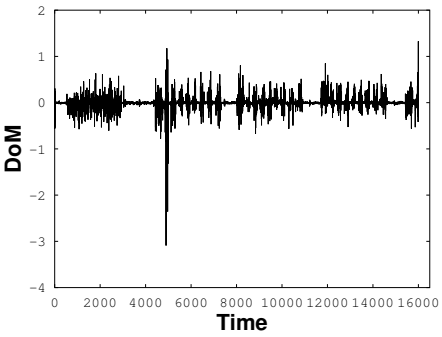


Fig. 1. Difference between the cluster centroids (y-axis) for the correct key of 1DES_wo_ctrmsr dataset described in Section 6

used for successful side-channel attacks (for a general overview see e.g. [19]). Nowadays, every device used for the applications of embedded security e.g. a smart card, an RFID tag, a mobile phone is considered to be a suitable target for the side-channel attackers to recover the secret key in indirect way. Successful attacks were performed on some (still) widely used commercial devices with security functionalities such as the KeeLoq-based remote keyless entry systems, the contactless Mifare DESFire card and the most recent attack on Atmel CryptoMemory cards [2].

In the late 90's Kocher et al. showed that, by measuring the power consumption of a smart card, one can retrieve information about the secret keys inside a tamper-proof device. The main observation is that the device's power consumption depends on the data being processed. This makes power traces correlated to certain intermediate variables, which, when directly depending on some key bits and some known data e.g. plaintext, allow for key recovery by simply using basic statistical tools such as the Distance of Means (DoM) test. This original approach is called Differential Power Analysis (DPA) [15]. The details of the DoM test can be found in [15], but as a general description, DoM considers a grouping (clustering) of the collected power traces for each candidate key and then selects the correct key (best clustering) when a clear peak is observed in the difference between the cluster centroids. DoM distinguisher is illustrated in Figure 1, where (in the y axis) the difference between the cluster centroids for the correct key is presented (we used 1DES_wo_ctrmsr dataset described in Section 6). The x axis contains the features (which is time in our application context, since power traces are collected over discrete time intervals).

Other known distinguishers include Pearson correlation coefficient (Correlation Power Analysis (CPA) [6]), Spearman's rank correlation [3], Mutual Information Analysis (MIA) [13] etc. Ideas from unsupervised learning were used for a distinguisher based on cluster analysis, so called Differential Cluster Analysis (DCA) [4]. CPA is still considered the first choice of the attacker, especially in the cases where power consumption is linearly dependent on Hamming weight (or distance) of the data. Although Pearson correlation solely tests for equidistance of cluster centroids, i.e. linearity, this limitation is proven to be not so restrictive in most of the devices used today [3].

Table 1. Analogies between DPA and Clustering

DPA with DoM distinguisher	Clustering
Possible Subkeys	Each subkey corresponds to a clusterings of Power Traces (in two groups)
DoM measure	Difference between cluster centroids
Correct Key≡ Peak in DoM	Best clustering ≡ Largest absolute value distance between centroids in just one or for a small number of features

2.2 DPA as Clustering

The acute reader may have already observed certain analogies between DoM based DPA (i.e. DPA with the Distance of Means distinguisher) and clustering. The analogy is based on the fact that the DoM distinguisher defines a clustering (in two groups) of the set of power traces for each possible key and consequently determines the secret key using a quality measure that is based on the difference between the two cluster centroids. More precisely, the secret key (equiv. best clustering) is determined by the largest peak in the difference between the cluster centroids, i.e. the largest absolute value distance (between the cluster centroids) in a small number of features. The analogies between DPA (with DoM distinguisher) and clustering are summarized in Table 1.

Up to this point, the clustering perspective of DoM based DPA is simply a "terminology rephrase" and the novel algorithmic insights that it provides may not be directly evident. As we will analyze lateron in detail, the power of the clustering formulation of DPA can be exploited in the cases where noise prevents DoM from identifying the correct key. The possible failures of DoM can be attributed to input data noise or to the use of countermeasure techniques that aim in hiding the key-relevant information from the power consumption. In the presence of high noise levels (that can be possibly due to the use of countermeasures), the input data must be appropriately pre-processed/analyzed such that the DoM distinguisher identifies the correct key. This can be achieved if we appropriately identify the structure of the signal that we wish to preserve in the data and the structure of the noise that we wish to remove.

Based on the clustering perspective of DPA we can define the following noise/signal formulation:
Signal→Clustering structures that "depend" on few features
Noise→Clustering structures that "depend" on many features

Intuitively the term "depend on few features" attempts to capture the fact that the DoM distinguisher uses the peak of the absolute value difference between the two cluster centroids *in a small subset of the available features*. Formally the term "depend on few features" is defined in Section 3.3 and it is taken to mean that the clustering objective does not change if we remove many features.

Using the clustering perspective of DPA and the afore noise/signal formulation we can state that the goal of a successful data mining algorithm in this application context is *to identify the clustering structures that "depend" on few features, or analogously to remove the clustering structures that "depend" on many features*. Naturally, due to

noise or data-structure reasons, there may exist other clusterings that "depend" on a small number of features but are not relevant to the correct key. However, as the empirical results verify in Section 6 the effective identification of these clustering structures (even at the potential cost of including non key-relevant clusterings) can enhance the effectiveness of the DoM method.

Having illustrated the clustering perspective of DPA and its potential utility for identifying the correct key, we will move on to illustrate the relevance of low-order localized eigenvectors of PCA for identifying the low-level signal within the data i.e. the relevant clustering structures that "depend" on a small number of features.

3 PCA, Localization and Clustering

3.1 PCA and Clustering

PCA is a very popular data preprocessing technique that is commonly understood as a data-reduction/approximation method. In principle, PCA maximizes the data variance in the reduced space and works by projecting the data matrix to the principal components (dominant eigenvectors) of the feature-covariance matrix. The intimate connections between PCA and clustering (often referred to as projected clustering, or subspace clustering), have been studied by several authors (such as [10], [20]).

In order to clarify the connections between PCA and clustering we briefly illustrate the results of [10]. Let X denote an input matrix that is $n \times m$, where n is the number of instances (power traces in our context) and m is the number of features (time scale in our context) and let also X_{fc} denote the feature centered data matrix (i.e. from each column of X the mean value is subtracted). If we consider the Singular Value of $X_{fc} = U\Sigma V^T$ it is easy to verify the following:

- The sample Covariance matrix can be written as: $Cov = \frac{1}{n-1} X_{fc}^T X_{fc}$.
- The right singular vectors of X_{fc} (columns of matrix V) are also the eigenvectors of matrix Cov.
- The singular values σ_i of X_{fc} are equal to $\sigma_i = \sqrt{\lambda_i (n-1)}$ where λ_i denotes the i^{th} eigenvalue of the sample covariance matrix Cov.

The main result of [10] states that the dominant left singular vector u_1 of X_{fc} can be regarded as a "continuous" clustering solution to the $K = 2$-means clustering problem (i.e. the elements of u_1 can be regarded as continuous approximations to the discrete cluster assignments of the instances). The connection to PCA projections becomes evident if we write u_1 as a projection of the input matrix X_{fc} to the dominant eigenvector of the sample Covariance matrix: $u_1 = X_{fc} v_1 / \sigma_1 = X_{fc} v_1 / \sqrt{(n-1)\lambda_1}$, where v_1 is the dominant right singular vector and σ_1 and λ_1 are as defined in the afore enumeration. One can obtain a discrete cluster solution using various discretization strategies of u_1. A simple discretization approach is to assign all instances i with $u_1(i) > mean(u_1)$ to cluster A and the rest to cluster $\overline{A}$. As it is rigorously analysed in [10] this simple process can be regarded as a continuous approximation solution to the $(K = 2)$-means objective function. Analogous results are also obtained for $K > 2$.

The results of [10] regarding the relationships between PCA and $(K = 2)$-means can be summarized in the following two lemmas:

Lemma 1 (Continuous Clustering Solution, equation 5, Theorem 2.2 in [10]). *A continuous clustering solution of the ($K = 2$)-means clustering problem can be derived by the dominant left singular vector of the feature-centered input matrix X_{fc}.*

Lemma 2 (Quality of the Continuous Clustering Solution, equation 6, Theorem 2.2 in [10]). *The quality of the continuous clustering solution is estimated (in a lower bound sense) by the dominant singular value of the feature-centered input matrix X_{fc}.*

Analogously to the afore lemmas, we can consider that the rest (low-order) left singular vectors of the feature-centered input matrix X_{fc} provide us with approximate clusterings of lower quality (with respect to the K-means objective), since they correspond to smaller singular values.

As we have analyzed in the previous section, in DoM based DPA we are interested in finding the 2-way clustering structure of the power traces that corresponds to the secret key. The relevance of PCA is not immediately evident since the dominant eigenvector(s) of PCA correspond to clustering structures that optimize the K-means clustering objective. This objective employs equally all the features and is not relevant to the criterion that is used for identifying the secret key. In the subsequent sections, the relevance of low-order PCA eigenvectors will be illustrated. These eigenvectors will correspond to low-quality (or at least non-optimal) clustering structures (according to the K-means objective) but will have certain properties (localization in a small number of features) that are crucial for identifying the secret key of the cryptographic device.

3.2 Eigenvector Localization

The key property that will allow us to detect the key-relevant clustering structures is localization. The term "eigenvector localization" is used to refer to cases where the majority of the mass of an eigenvector is located in a small number of entries (i.e. most of the eigenvector entries are zero and near-zero). This phenomenon has been observed in several diverse application areas such as DNA single-nucleotide polymorphism data, spectral and hyperspectral data in astronomy (see [8],[7],[18] and references therein). Eigenvector localization is commonly measured using the *Inverse Participation Ratio* [8],[7] that is defined as:

$$IPR(v) = \sum_{i=1}^{m} v(i)^4 \tag{1}$$

Where v is the eigenvector (of length m) whose localization we want to measure. It should also be noted that this definition assumes that the eigenvectors are normalized (i.e. $\sum_{i=1}^{m} v(i)^2 = 1$). The higher the value of IPR the more localized the eigenvector is.

It can be observed that IPR is related to the fourth moment in statistics and can effectively measure the level of concentration of the eigenvector values. For example if the values of an eigenvector are equally scattered in all its indexes, i.e. $v = (\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}, ..., \frac{1}{\sqrt{m}})$ then the IPR will be $IPR(v) = 1/m$. While, in the extreme case where there is only one non-zero entry, i.e. $v = (0, .., 1, ..., 0)$ the IPR value will be $IPR(v) = 1$. Naturally, localization can be quantified with various other measures, such as the simple sum of

absolute values. However, in the context of this work we focus solely on the IPR measure since it is consistent with the relevant work and also works effectively in practice (as observed in experiments Section 6). A deeper study on the different potential measures for quantifying eigenvector localization constitutes an interesting topic for further work.

3.3 Localized Principal Eigenvectors and Clustering Structures That Depend on Few Features

In Section 3.1 we have illustrated the relevance of PCA and its dominant eigenvectors for obtaining (continuous) approximate clusterings that optimize the K-means objective. We will now move on and show that the projection of a (feature centered) data matrix to a low-order localized eigenvector corresponds to a (continuous) clustering solution that has lower quality (with respect to the K-means objective) and "depends" only on a small number of features.

Intuitively, the term "depends on a small number of features" means that the quality of the clustering does not change if we remove the features where the eigenvector is localized. This is formally shown in the following proposition:

Proposition 1. *Let X_{fc} be an $n \times m$ matrix (n=# instances, m=# features). If σ_l, u_l and v_l are the l^{th} singular value and vectors of X_{fc} with $v_l(i) = 0$ for all $i \in A \subset \{1,...,m\}$, then σ_l,u_l and v_l will also be singular value and vectors of $X_{fs} = X_{fc}\mathbf{diag}(u)$ where $u(i) = 0$ for all $i \in A$, $u(i) = 1$ for all $i \in \overline{A}$ and $\mathbf{diag}(u)$ denotes a diagonal matrix with vector u in its diagonal.*

Proof. In our proof we will initially show that v_l is an eigenvector of $X_{fs}^T X_{fs}$ and that u_l is an eigenvector of $X_{fs}X_{fs}^T$.

We have that $X_{fs}^T X_{fs} v_l = \mathbf{diag}(u) X_{fc}^T X_{fc} \mathbf{diag}(u) v_l$. Now since $v_l(i) = 0$ and $u(i) = 0$ for the same i, we have that $\mathbf{diag}(u)v_l = v_l$. Thus we can write:
$\mathbf{diag}(u)X_{fc}^T X_{fc}\mathbf{diag}(u)v_l = \mathbf{diag}(u)X_{fc}^T X_{fc} v_l$. Now since v_l is a right singular vector of X_{fc} we can derive that:
$\mathbf{diag}(u)X_{fc}^T X_{fc} v_l = \mathbf{diag}(u)X_{fc}^T(\sigma_l u_l) = \sigma_l \mathbf{diag}(u)X_{fc}^T u_l = \sigma_l^2 \mathbf{diag}(u)v_l = \sigma_l^2 v_l$

Above, we have shown that v_l is an eigenvector of $X_{fs}^T X_{fs}$ with corresponding eigenvalue σ_l^2 and thus v_l is also a right singular vector of X_{fs} with corresponding singular value σ_l.

Now for u_l we can write $X_{fs}X_{fs}^T u_l = X_{fc}\mathbf{diag}(u)\mathbf{diag}(u)^T X_{fc}^T u_l = X_{fc}\mathbf{diag}(u)(\sigma_l v_l) = \sigma_l X_{fc}\mathbf{diag}(u)v_l = \sigma_l X_{fc} v_l = \sigma_l^2 u_l$

Above, we have shown that u_l is an eigenvector of $X_{fs}X_{fs}^T$ with corresponding eigenvalue σ_l^2 and thus u_l is also a left singular vector of X_{fs} with corresponding singular value σ_l.

Since u_l and v_l correspond to the same singular value they will effectively be a singular vector pair of X_{fs} with singular value σ_l. **QED**

Informally the afore proposition can be summarized, using the same notation, as follows:

- Let X_{fc} be an input matrix with a localized low-order right singular vector v_l (recall that this is also an eigenvector of the feature-covariance matrix).
- The corresponding clustering solution will be u_l, the l^{th} left singular vector of X_{fc} or equivalently, the projection of the input data matrix to v_l.
- The (continuous) quality of this clustering solution will be equal to the l^{th} singular value of X_{fc}.
- If we apply feature selection and remove the features where v_l is localized (in the Proposition this matrix is denoted as $X_{fs} = X_{fc}\mathbf{diag}(u)$), then the matrix X_{fs} will have σ_l, u_l and v_l as signular values and vectors.
- This effectively means, that the quality σ_l of the (continuous) clustering u_l is the same for both X_{fc} (original input matrix) and X_{fs} (matrix after feature selection).

The analysis presented in this Section should have clarified the utility of localized eigenvectors of a feature covariance matrix for identifying clusters that "depend on few features". However, one question that still remains is *Why (and when) should we expect these localized eigenvectors to appear in the spectrum of a Covariance matrix*. This is a valid and important question since an $n \times m$ matrix can at most have $min(n,m)$ singular vectors. These can be much less than the number of possible clusterings of the n instances.

4 Why (and when) Are the Eigenvectors of PCA Localized

We should initially state that our motivation to study localized eigenvectors in the context of DPA was based on [5] where it was demonstrated that low-order localized eigenvectors do appear in the spectrum of the Covariance matrix and also carry important information related to the secret key used by the cryptographic device. Based on these findings we will study in this section a simple model that can explain the appearance of these useful, cluster related localized eigenvectors.

Our model considers a high noise "unstructured" matrix and a low-signal rank-1 update that "inserts" the relevant clustering to the unstructured data matrix. The terms "high-noise" vs. "low-signal" are employed to stress that the singular values of the "unstructured matrix" are larger than the singular value of the rank-1 update. This eventually means that these cluster structures would be "buried" in the low-order spectrum of the resulting data matrix and would not be detectable using a standard spectral clustering technique.

In mathematical terms our model can be stated as:

$$X_{input} = X_{unstruct} + \gamma u_{cl} v_{cl}^T \tag{2}$$

Where X_{input} and $X_{unstruct}$ are (instance$\times$variable) matrices ($n \times m$), u_{cl} is an $n \times 1$ vector that contains the instance (power trace) clustering structure[1], v_{cl} is the localized $m \times 1$ vector with $||v_{cl}||_2 = 1$ that contains only a small number of non-zero indexes (i.e. it is localized) and γ is the "power" of the signal that is small as compared to the largest

[1] $u_{cl}(i) = \sqrt{n_2/nn_1}$ if i belongs to cluster A_1 and $u_{cl}(i) = -\sqrt{n_1/nn_2}$ if i belongs to cluster A_2. n_1 and n_2 are the cluster sizes of A_1 and A_2 respectively and $n = n_1 + n_2$ is the total number of instances.

eigenvalues of $X_{unstruct}$. Based on this model, we will inspect whether the localized right singular vectors of X_{input} can be effectively used for detecting the "hidden" clustering structure u_{cl} that depends on a small number of features.

It can be observed that the spectrum of X_{input} will be determined by the correlation of u_{cl} to the left singular vectors of $X_{unstruct}$ and also by the correlations of v_{cl} to the right singular vectors of $X_{unstruct}$. In our simulations, we will consider that the structures u_{cl} and v_{cl} are not already present in the unstructured data matrix. This is achieved by considering that the left and right singular vectors of $X_{unstruct}$ are random orthogonal matrices (Haar distribution) [22,9], thus the vectors u_{cl} and v_{cl} will exhibit a (uniformly) random correlation with the singular vectors of $X_{unstruct}$. Moreover, in order to simulate the $X_{unstruct}$ matrix, we employ the actual singular values that were observed in the dataset 1DES_RandomDelays, which is described in the Experiments Section 6.

Based on the above, we define $X_{unstruct}$ as follows:

$$X_{unstruct} = U_{un} \Sigma V_{un}^T$$

where U_{un} and V_{un}^T are uniform random orthogonal matrices [22,9] and Σ contains the singular values of dataset 1DES_RandomDelays.

Based on this definition of $X_{unstruct}$ we have experimented with the two parameters of the model, the γ parameter and the number of features that determine the cluster structure (i.e. the number of zeros in v_{cl}). The simulations illustrate that when the size of γ is "sufficiently large" and also when the cluster structure depends on "sufficiently few" features, we can effectively use the IPR measure for detecting the hidden clustering structure. The terms "sufficiently large" and "sufficiently few" are explored within the experiments[2].

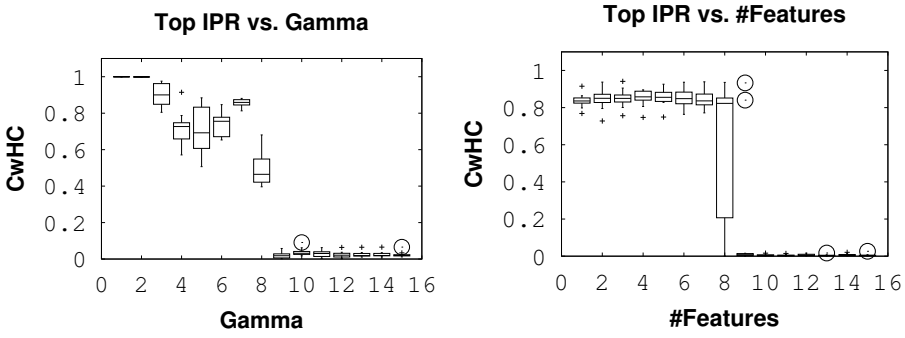
In Figure 2(a) we have fixed the level of localization (i.e. number of zeros in the in v_{cl} of formula 2), and we have used multiple γ values, while in Figure 2(b) we have fixed γ and have varied the localization of the hidden clustering structure. More precisely, in Figure 2(a) we have fixed v_{cl} at 10 Features, and the initial value for the γ parameter was taken to be equal to the maximum singular value of $X_{unstruct}$ divided by 2. Consecutively, in each run we have divided γ by 2, i.e. in the 8^{th} run we set γ to be equal to the maximum singular value of $X_{unstruct}$ divided by 2^8.

In Figure 2(b), we have fixed the γ value to be equal to the maximum singular value of $X_{unstruct}$ divided by 2^7 and we have taken the initial number of non-zero element in v_{cl} to be 60. Consecutively, in each run we have set the number of features as multiples of 60, i.e. in the 5th run the number of features (non-zero elements) in v_{cl} are taken to be $5 \cdot 60$.

In both Figures 2(a),2(b), in order to detect the hidden clustering structure, we have executed the following procedure:

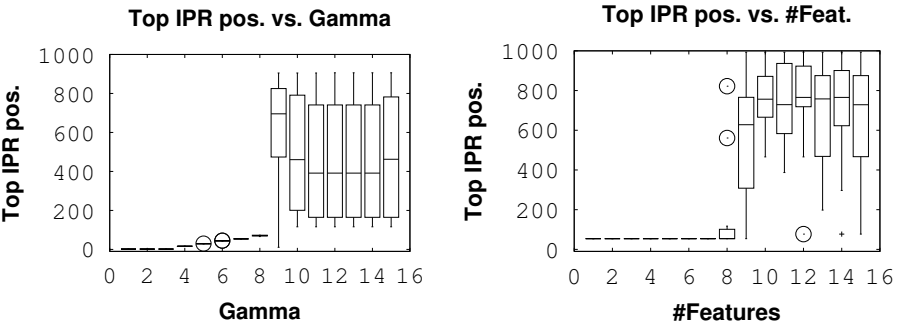
- We compute the SVD of $X_{input} = U_{input} \Sigma_{input} V_{input}^T = X_{unstruct} + \gamma u_{cl} v_{cl}^T$

[2] In order to facilitate the reproduction of empirical results, the relevant model simulation code is available at the website of the correspondence author: http://sites.google.com/site/mavroeid/



(a) IPR vs. Gamma (b) IPR vs. # Features

Fig. 2. Effectiveness of top IPR vector for detecting hidden clustering structure



(a) Position of top IPR vs. Gamma (b) Position of top IPR vs. # Features

Fig. 3. Position of top IPR vector

- We compute the IPR of the right singular vectors (columns of V_{input})
- We select the left singular vector that corresponds to the top IPR right singular vector and measure its correlation with the hidden clustering structure using the inner product. (*CwHC* measure, x axis in Figure 2)

The afore procedure is repeated 10 times and Figure 2 reports the resulting boxplots. Interestingly, Figure 2 illustrates that even when we insert a very low-signal clustering structure (i.e. when γ is 2^7 times smaller than the maximum singular value of $X_{unstruct}$), that is localized with less than 10% of the available features, IPR can effectively detect the hidden clustering structure.

Figure 3 illustrates the positions (ordered according to singular values) of the top IPR vectors that are used for finding the hidden clustering structure in Figure 2. Again boxplots are presented based on the 10 simulations of X_{input}.

5 Related Work

To the extent of our knowledge, low-order localized eigenvectors of graph adjacency and Laplacian matrices have received little attention within the data mining community ([8,7] and references therein). As compared to these approaches the main distinctive characteristic of our work is that it focuses on an (instance$\times$feature) input matrix and interprets localized low-order eigenvectors as clusterings that depend on a small number of features.

Recently, there were a few works showing the potential of PCA for side-channel analysis [1,5]. The impact of low-order eigenvectors in attacking smart card platforms was demonstrated in experiments but the theoretical reasoning on the effectiveness of the new method was lacking [5]. Our work fills in this gap in proving that the previous study was not an isolated and random experiment but there is indeed a lot of potential in exploring similar techniques that are already known for machine learning researchers.

6 Experiments

6.1 Data Description

To perform side-channel analysis some hardware equipment is required, being part of a measurement set-up. A typical set-up for measurements includes a smart card (on which the target algorithm is implemented), a smart card reader, an oscilloscope for the acquisition of power traces and a PC for the analysis and key recovery. All our experiments are performed using a programmable card from Atmel (ATMega163+24C256). The card contains an Atmel 8-bit AVR RISC-based microcontroller that combines 16KB of programmable flash memory, 1KB SRAM and 512B EEPROM. Single and Triple DES implementations that we considered were all software implementations, with or without countermeasures. An implementation of triple DES consists of three DES algorithms where the new key is 112 bits long because the first and last DES have the same key.

The first datasets called 1DES_wo_ctrmsr and 3DES_wo_ctrmsr are single DES and triple-DES implementations without countermeasures. However, cryptographic algorithms are in practice often implemented with one or more countermeasures rendering side-channel analysis. The countermeasures are commonly divided into two groups: masking or hiding [19]. Masking countermeasures are based on masking data or the key (or both) by adding a random value to the input that is created to make the intermediate variables data- and key-independent complicating in this way DPA substantially. Hiding countermeasures aim at making a side-channel e.g. power consumptions unrelated to the data processed for example by burying the signal into noise. For the measurements of simple DES with countermeasures we employed random delays (1DES_RandomDelays), which introduces random wait states during execution of the algorithm. In this way the effect of misaligned traces is obtained.

We have also employed data masking countermeasures 3DES_DataMasking for triple DES, so the inputs are masked with a random value and unmasked at the end to get the correct output.

As our experiments use DES and triple DES, we give some details of the algorithms next. The Data Encryption Algorithm (DES) was invented in the 70's by IBM and used

Table 2. Datasets used in experiments. In order to facilitate the reproduction of empirical results, all datasets are available at the website of the correspondence author: http://sites.google.com/site/mavroeid/

1DES_wo_ctrmsr	DES without countermeasures
1DES_RandomDelays	DES with random delays countermeasure
3DES_wo_ctrmsr	3DES without countermeasures
3DES_DataMasking	3DES with data masking countermeasure

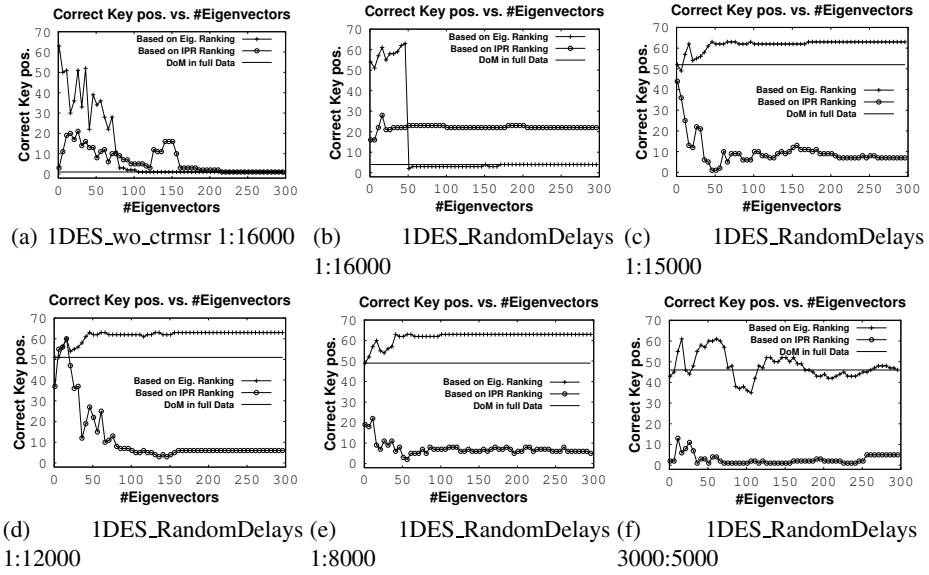
as the main encryption standard for more than two decades. Due to the short key, DES is nowadays mainly used as a part of the triple DES algorithm (3DES), but nevertheless attacking DES remains the first step in the side-channel analysis of 3DES.

DES uses a 64-bit key and it operates on 64-bit blocks of plaintext as input and returns blocks of 64 bits as output (called ciphertext) after 16 rounds. Each round has several operation, but the most important one is the non-linear function f that contains 8 substitution boxes $(S_1,\ldots,S_8)$, so-called S-boxes. The S-boxes have a 6-bit input, and a 4-bit output derived by a table look-up. The most important property for side-channel analysis is that the 6-bit inputs, and hence also 4-bit outputs of S-boxes, depend only on plaintexts and 6 bits of the key. In other words, knowing plaintexts and making a hypothesis on the 6-bit subkeys, we can monitor only one S-box to learn the right hypothesis on the subkey (out of 2^6 possibilities). By monitoring, we mean computing the S-box outputs and a function of the inputs, hereafter called the selection function as in [15] (for known plaintexts and a key guess) and correlating the values to the power consumption measured during this computation. The reasoning behind lies in the simple fact that power consumed by a device depends on the data being processed.

The procedure for the data acquisition and the key recovery, consists of several steps explained below. We use the example of DES to list the steps.

- Denote the selection function by $f(x,k)$ where k is typically a small part of the key (here 6 bits). For a large number (say n) of randomly generated plaintexts, we can compute the value of f for each 6 bits of a subkey. At the same time we collect power measurements of the device while performing the algorithm.
- Each power trace contains the values at m time points i.e. samples. Hence, we can create a matrix A of size nxm that contains the power traces corresponding to different plaintexts.
- Next step is to calculate the hypothetical values of f for every possible subkey k. In the case of DES there are 64 different subkeys for each S-box. As the selection function we simply choose one bit of the S-box output e.g. MSB (or LSB) so the function $f(x,k)$ can be either 0 or 1. Based on this value we sort all power traces into two sets, say S_0 and S_1. Note that this will result in a power-trace clustering for each possible subkey.
- We apply distance of means (DoM) test to the S_0 and S_1 partition plotting each so-called differential traces that is computed as:

$$d = \overline{S_1} - \overline{S_0}.$$



(a) 1DES_wo_ctrmsr 1:16000 (b) 1DES_RandomDelays 1:16000 (c) 1DES_RandomDelays 1:15000

(d) 1DES_RandomDelays 1:12000 (e) 1DES_RandomDelays 1:8000 (f) 1DES_RandomDelays 3000:5000

Fig. 4. Correct Key Position based on DoM method for single-DES

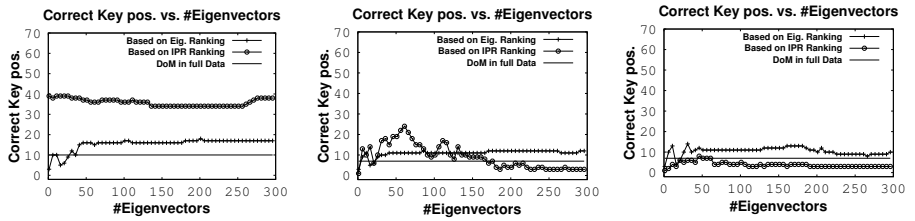
Out of 64 differential traces we select the one with the highest peaks at certain time sample as the correct key.

6.2 Empirical Results

In order to evaluate the effectiveness of IPR and localized eigenvectors for retrieving the correct key, we adopt the following procedure:

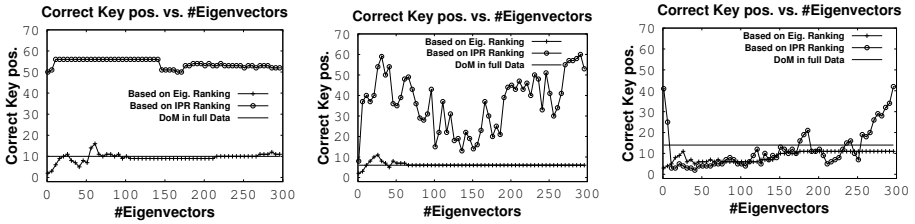
- We compute the SVD of the centered input $n \times m$ (power trace $\times$ time) matrix $X_{fc} = U\Sigma V^T$.
- We compute the IPR of the right singular vectors (i.e. the columns of V).
- We compute projections based on PCA and IPR. I.e. We compute $Pr_{(PCA)}(k) = X_{fc}V_{1:k}V_{1:k}^T$ and $Pr_{(IPR)}(k) = X_{fc}V_{IPR(1:k)}V_{IPR(1:k)}^T$, where $V_{1:k}$ contains the dominant right singular vectors (that correspond to the largest singular values), while $V_{IPR(1:k)}$ contains the top IPR singular vectors.
- We compare the results obtained by the DoM measure in the full data, in $Pr_{(PCA)}$ and $Pr_{(IPR)}$. I.e. for each input matrix we compute the DoM measure and we rank the keys according to the largest observed peak. Since we know for these datasets the correct key, we can observe whether the IPR or PCA preprocessing improves the position of the correct key.

In Figure 4 we evaluate the effectiveness of IPR in single-DES datasets (1DES_wo_ctrmsr, 1DES_RandomDelays). The y-axis of Figure 4 reports the position of the correct key, based on the full dataset and also using PCA and IPR preprocessing.



(a) 3DES_wo_ctrmsr 1:16000 (b) 3DES_wo_ctrmsr 1:12000 (c) 3DES_wo_ctrmsr 2900:84000

Fig. 5. Correct Key Position based on DoM method for 3DES_wo_ctrmsr



(a) 3DES_DataMasking 1:16000 (b) 3DES_DataMasking 1:15000 (c) 3DES_DataMasking 1:12000

Fig. 6. Correct Key Position based on DoM method for 3DES_DataMasking

The optimal result is to obtain a *position=1* for the correct key. Moreover, knowing that the largest peak in these datasets (at the aligned data) occurs around time 4900 (in accordance to DPA terminology, these are called interesting points), we have investigated the effect of IPR and PCA as we zoom to time 4900 (the original time scale is 1-16000).

In the single-DES dataset without countermeasures 1DES_wo_ctrmsr (Figure 4(a)), the full-data matrix already retrieves the correct key in top-position. Moreover, we can observe that PCA and IPR preprocessing converge rather quickly to position=1 with PCA achieving a faster convergence while IPR being more effective when a very small number of eigenvectors are used.

In the single-DES dataset with random delays 1DES_RandomDelays (Figures 4(b), 4(c), 4(d), 4(e), 4(f) for different time-intervals), the full-data matrix fails to retrieve the correct key and in most cases (with the exception of the 1-16000 time interval) ranks it very low. On the other hand IPR is very effective in the majority of experiments with the notable exception of the 1-16000 time interval. More precisely, we can observe that the performance of IPR dramatically improves as we zoom to time 4900 (interest-point). Notably, when we focus on the 3000-5000 time scale IPR consistently retrieves the correct key in top positions.

In the triple-DES dataset without countermeasures DES_wo_ctrmsr (Figures 5(a), 5(b), 5(c)), the full-data matrix places the correct key in the 10th position.We can again

observe that the IPR ranking requires an adequate zoom into the interest-point for effectively identifying the correct key. A similar behaviour is also observed in triple-DES with Data Masking.

7 Discussion

In this work we have presented a thorough theoretical framework that illustrates the relevance of low-order localized eigenvectors in the application context of DPA. In our analysis, we have considered that the key-relevant information is modelled as a "low-signal", localized pattern that is embedded in a "high-noise" dataset. In this respect our results generalize beyond DPA and are applicable to analogous low-signal/hidden pattern problems.

Based on the empirical results one can identify two interesting further work topics that can potentially enhance the practical effectiveness of our framework. The first is related to the automatic determination of the appropriate number of eigenvectors that are needed in order to optimize the position of the correct key. The successful tackling of this issue requires further statistical analysis of the IPR measure in order to be able to identify statistically significant localized eigenvectors and not simple noise artifacts.

The second issue is related to the determination of the appropriate time-interval that maximizes the effectiveness of IPR and localization. As discussed above, to maximize the effect of IPR, we rely on the ability of "zooming" into the traces i.e. we assume the points with maximal leakage (also sometimes called interesting points) being known to some accuracy. This is a standard assumption for DPA, and there are several methods proposed in the relevant literature. For example, an attacker can simply apply the correlation with input data analysis [11]. Similarly, one can use template attacks [1] to learn more about the points of interest. It constitutes an interesting issue for further work to explore the required tightness of the time-interval that can guarantee the effectiveness of IPR and localization.

References

1. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template Attacks in Principal Subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)
2. Balasch, J., Gierlichs, B., Verdult, R., Batina, L., Verbauwhede, I.: Power Analysis of Atmel CryptoMemory – Recovering Keys from Secure EEPROMs. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 19–34. Springer, Heidelberg (2012)
3. Batina, L., Gierlichs, B., Lemke-Rust, K.: Comparative Evaluation of Rank Correlation Based DPA on an AES Prototype Chip. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 341–354. Springer, Heidelberg (2008)
4. Batina, L., Gierlichs, B., Lemke-Rust, K.: Differential Cluster Analysis. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 112–127. Springer, Heidelberg (2009)
5. Batina, L., Hogenboom, J., van Woudenberg, J.G.J.: Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 383–397. Springer, Heidelberg (2012)

6. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
7. Cucuringu, M., Blondel, V.D., Van Dooren, P.: Extracting spatial information from networks with low-order eigenvectors. CoRR, abs/1111.0920 (2011)
8. Cucuringu, M., Mahoney, M.W.: Localization on low-order eigenvectors of data matrices. CoRR, abs/1109.1355 (2011)
9. Diaconis, P., Shahshahani, M.: The subgroup algorithm for generating uniform random variables. Probability in the Engineering and Informational Sciences 1(01) (1987)
10. Ding, C.H.Q., He, X.: *K*-means clustering via principal component analysis. In: ICML (2004)
11. Fahn, P.N., Pearson, P.K.: IPA: A New Class of Power Attacks. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 173–186. Springer, Heidelberg (1999)
12. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic Analysis: Concrete Results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
13. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
14. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
15. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
16. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
17. Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4) (2007)
18. Mahoney, M.W.: Randomized algorithms for matrices and data. Foundations and Trends in Machine Learning 3(2), 123–224 (2011)
19. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. In: Advances in Information Security. Springer-Verlag New York, Inc., USA (2007)
20. Meinicke, P., Ritter, H.: Local pca learning with resolution-dependent mixtures of gaussians. In: Ninth International Conference on Artificial Neural Networks, ICANN 1999. (Conf. Publ. No. 470), vol. 1, pp. 497–502 (1999)
21. Quisquater, J.-J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, p. 200. Springer, Heidelberg (2001)
22. Stewart, G.W.: The efficient generation of random orthogonal matrices with an application to condition estimators. SIAM Journal on Numerical Analysis 17(3), 403–409 (1980)

Classifying Stem Cell Differentiation Images by Information Distance

Xianglilan Zhang*, Hongnan Wang, Tony J. Collins,
Zhigang Luo, and Ming Li**

School of Computer, National University of Defense Technology,
Changsha, Hunan 410072, China
David R. Cheriton School of Computer Science, University of Waterloo,
Waterloo, Ontario N2L 3G1, Canada
Stem Cell and Cancer Research Institute, Mcmaster University,
Hamilton, Ontario L8S 4K1, Canada
{x322zhang,mli}@uwaterloo.ca,
WangHongnan1979@yahoo.com,
tjc@mcmaster.ca,
zgluo@nudt.edu.cn

Abstract. The ability of stem cells holds great potential for drug discovery and cell replacement therapy. To realize this potential, effective high content screening for drug candidates is required. Analysis of images from high content screening typically requires DNA staining to identify cell nuclei to do cell segmentation before feature extraction and classification. However, DNA staining has negative effects on cell growth, and segmentation algorithms err when compound treatments cause nuclear or cell swelling/shrinkage. In this paper, we introduced a novel Information Distance Classification (IDC) method, requiring no segmentation or feature extraction; hence no DNA staining is needed. In classifying 480 candidate compounds that may be used to stimulate stem cell differentiation, the proposed IDC method was demonstrated to achieve a 3% higher F_1 score than conventional analysis. As far as we know, this is the first work to apply information distance in high content screening.

Keywords: information distance, stem cell differentiation image classification, compound classification.

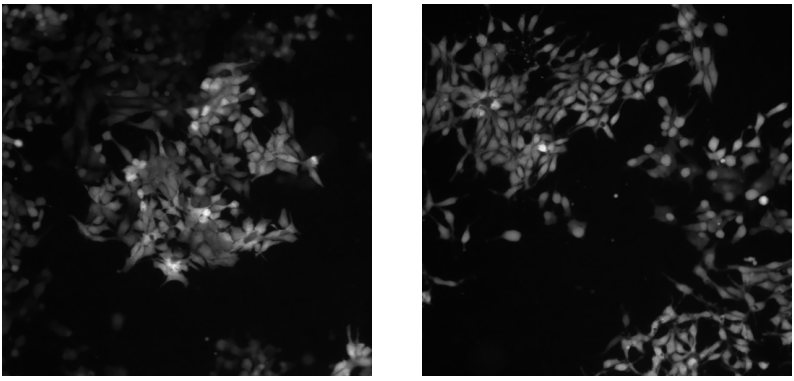
1 Introduction

Stem cells are characterized by their ability to transform, by a process referred to as differentiation, from a primitive pluripotent state into diverse specialized mature cell types, and to self-renew to produce more pluripotent stem cells [1]. Various tissues, such as muscles and nerves, are grown and transformed from stem cells. In current stem cell research, compounds that will induce this differentiation are being sought

* Xianglilan Zhang is currently a visiting Ph.D. student at David R. Cheriton School of Computer Science of University of Waterloo (from October 2010 to October 2012).
** Ming Li is the corresponding author.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 269–282, 2012.

often through high content screening. High content screening, whereby thousands of compounds are tested for the effect on cells, is a key approach to the identification and development of chemical-based therapeutics that serve as tools for replacement therapy and as novel drugs for the treatment of degenerative diseases.



(a) High Control Compound (b) Low Control Compound

Fig. 1. Cluster illustration. Each image represents a different type of cluster. Fig. 1 (a) shows the cell situation after a high control compound is added; Fig. 1 (b) shows the cell situation after a low control compound is added. Theoretically, the cells treated by a high control compound should become larger in size and decrease more in number than cells treated by a low control compound.

Primary high content screening is time-consuming and expensive, and secondary assays to validate the primary screening are also generally complex. This is especially true for stem cell biology where expansion of undifferentiated cells is technically more challenging. Thus, efficient and accurate computer-assisted compound classification is critical for efficient secondary biological compound identification. Our goal is to analyze images of human embryonic stem cells expressing a green fluorescent protein (GFP)-based pluripotency reporter and thus to identify those compounds that induce the cells to differentiate. Upon differentiation with a known differentiation treatment (BMP4; high control), the cells show reduced GFP fluorescence intensity, are reduced in numbers, and appear larger. As shown in Fig. 1, a high control compound means a known active compound that can cause stem cell differentiation; a low control compound is a known inactive compound that does not induce stem cell differentiation. If a compound tested is classified with the high control cluster, it can be treated as an active compound; if it is classified with the low control cluster, it can be treated as an inactive compound. Through stem cell differentiation image classification, we can label the presumed active compounds, thus decreasing the number of active compound candidates. Activity can be confirmed with a dose-effect experiment. The more accurately and

efficiently active compounds are labeled, the cheaper and faster the biological verification step will be.

In the above assay, compounds are typically classified based on the GFP level in cells. In the classical approaches [3] (i.e., conventional univariate analysis and multivariate analysis), a DNA stain is added to label cell nuclei. Nuclear segmentation algorithms are then fine-tuned to identify individual nuclei, which are presumed to represent individual cells. These nuclei are then used as foci to identify and segment the cytoplasmic area. Most of the DNA stains are cytotoxic or at least cytostatic, and have a negative impact on cell growth. Moreover, the need to set constraints on the segmentation algorithms, such as maximum/minimum size, means segmentation errors when compound treatments cause nuclear or cell swelling/shrinkage.

Therefore, we introduce a novel method, the Information Distance Classification (IDC) method, to classify compounds, whereby we analyze the similarity of stem cell differentiation images from each well containing one unique compound. Unlike traditional methods, ours does not involve segmentation, and thus does not require DNA staining. The method depends on information distance, based on Kolmogorov complexity [4], first introduced by [5], and has been applied to many different areas, from image processing to weather broadcasting, to software engineering, and to bioinformatics [6–27]. This method skips the feature-extraction step in practice. Therefore, it does not need manual intervention, as required in conventional analysis, to define the cell count and brightness for compound classification. Using our IDC method, we achieve results comparable to those of conventional analysis. To the best of our knowledge, our paper is the first to utilize information distance for a novel application in high content screening.

2 Information Distance

The classification of stem cell differentiation images essentially depends on determining whether the cells in an image are "similar" to those acquired from high or low control compounds. Conventional methods involving segmentation to acquire cell information for extracting features is just one way to measure image similarity. We want to find a general and optimal approach for measuring the similarity between two cell images.

In the early 1990s, in [5], the authors studied the energy cost of conversion between two strings, x and y. John von Neumann hypothesized that performing 1 bit of information processing costs $1K_BT$ of energy, where K_B is the Boltzmann's constant and T is the room temperature. In the 1960s, observing that reversible computations can be done for free, Rolf Landauer revised von Neumann's proposal to hold only for irreversible computations. Starting from this von Neuman-Landauer principle, [5] proposed using the minimum number of bits needed to convert between x and y to define their distance. Formally, with respect to a universal Turing machine U, the cost of conversion between x and y is defined as:

$$E(x, y) = \min\{|p| : U(x, p) = y,\ U(y, p) = x\} \tag{1}$$

It is clear that $E(x, y) \leq K(x|y) + K(y|x)$. In [5] the following optimal result was obtained, up to an additive factor of $O(\log(|x| + |y|))$:

Theorem 1. $E(x, y) = \max\{K(x|y), K(y|x)\}$.

Here $K(x|y)$ is the Kolmogorov complexity [4] of a binary string x condition to another binary string y, $K(x|y)$, which is informally defined to be the length of the shortest program that outputs x with input y. We refer the readers to [4] for further details of Kolmogorov complexity and its rich applications in computer science and many other disciplines.

This complexity has enabled us to define the information distance between two sequences x and y as:

$$D_{\max}(x, y) = \max\{K(x|y), K(y|x)\}. \tag{2}$$

This distance is shown to satisfy the basic distance requirements such as positivity, symmetricity, and triangle inequality. Furthermore, $D_{\max}$ is "universal" in the following sense. A distance D is *admissible* if

$$\sum_{y} 2^{-D(x,y)} \leq 1. \tag{3}$$

$D_{\max}(x, y)$ satisfies the above requirement because of Kraft's Inequality (with the prefix-free version of Kolmogorov complexity). It was proved in [5] that for any admissible computable distance D, there is a constant c, for all x, y:

$$D_{\max}(x, y) \leq D(x, y) + c. \tag{4}$$

Thus, if any such distance D discovers some similarity between x and y, so will $D_{\max}$. Therefore, $D_{\max}$ is universal.

According to this theory, given two images x and y, the ultimate distance between them is $D_{\max}(x, y)$. That is, if there is another way of finding whether x and y are similar, then $D_{\max}$ finds it too. The only problem is that $K(x|y)$ is known to be uncomputable. Nevertheless, for practical applications, universal compression algorithms can be used to approximate $K(x|y)$:

$$D_{\max}(x, y) = \max\{C(x|y), C(y|x)\}. \tag{5}$$

where C is a compression algorithm.

This approach was first introduced by [20] to measure the information distance between two genomes. Many other applications in various different fields have been found. Recently, it was used in [8] and [21] to measure the similarity of images, and both studies obtained promising results.

3 Materials

In this section, we describe the image data acquisition process. Images from the OCRiT v1O4 primary screening of the NIH clinical compound collection were used to develop the stem cell differentiation analysis. As shown in Fig. 2, this collection equated to six library plates run in triplicate to become 18 assay plates. Each plate had 96 wells in

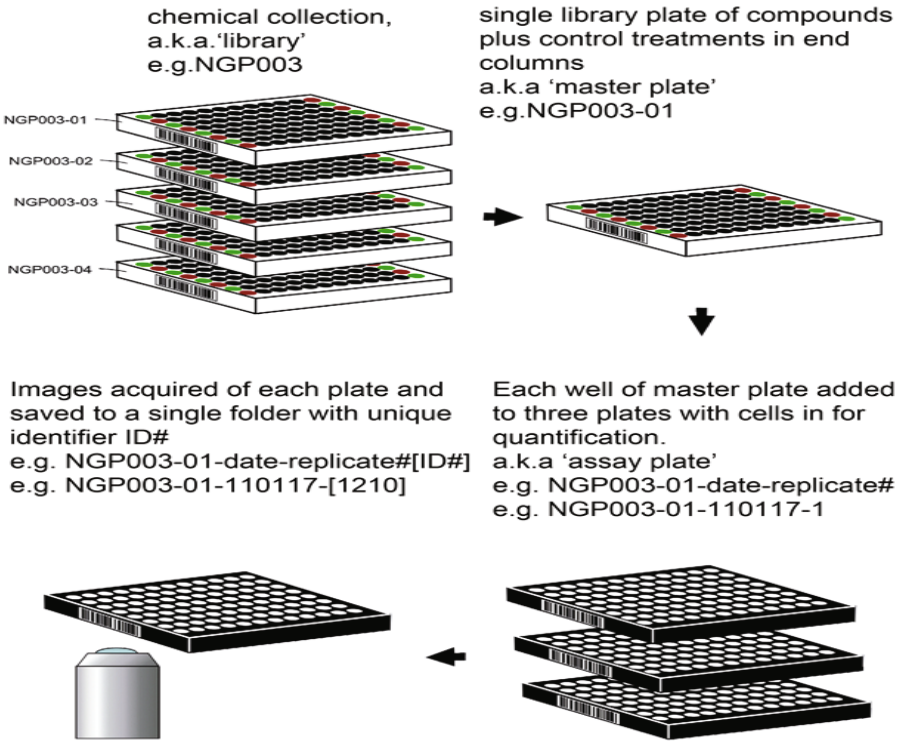


Fig. 2. Assay workflow. Single library plate contained three assay plates. Each plate had control treatments in its end columns. The images of each plate were captured and then stored in a single folder with a unique ID number.

total, and each well contained a different compound at 10μM concentration. Eight high and eight low control compounds were alternated in the columns at either end of the plates. Here, high control means known active, and low control means known inactive. Thus, each plate had 80 compounds tested.

Human stem cells expressing GFP pluripotency reporters were added to each well to show the cytoplasm of cells. At the end of the experiment, the DNA stain, Hoechst, was added to visualize the cell nuclei. Two fluorescent images (GFP and Hoechst) were acquired for nine fields per well. Since each compound was sampled three times, this totaled 27 images per compound.

It should be noted that, because IDC analysis does not require segmentation, only the GFP images were analyzed. Using IDC would have removed the need for any pre-acquisition sample processing apart from a media wash to remove the autofluorescent growth medium.

4 Information Distance Classification Method

Here, we propose our IDC method to describe the process of compound classification.

It is important to note that scattering GFPs over the cell culture wells causes uneven illumination and a "speckle" effect outside of the cells. To remove these illumination inconsistencies and noises in images, we change Eq. 5 to the following:

$$D_{\max}(x,y) = \min\{\max\{C(x|y), C(y|x)\}, \max\{C(f(x)|f(y)), C(f(y)|f(x))\}\}. \tag{6}$$

where

$$f(I) = \frac{I - \min(I)}{\max(I) - \min(I)}$$

$$\min_I\{\iint_\Omega \sqrt{I_x^2 + I_y^2}\mathrm{d}x\mathrm{d}y + \frac{1}{2}\iint_\Omega (I - I_0)^2\mathrm{d}x\mathrm{d}y\}$$

I is the denoising result of the total variation method [28]. Ω is an image domain. I_x and I_y are the first-order partial derivatives of image I with respect to x and y directions. I_0 is the original image.

Generally speaking, this method contains seven steps:

1. Use function f in Eq. 6 to do image preprocessing.
2. Use MPEG encoder to compress original and transformed images.
3. Use information distance to measure the distance between any pair of images.
4. Do information distance statistics on the high/low control images that represent high/low control compounds.
5. Calculate the average information distance between any image of the compound tested and the control images.
6. Classify the images.
7. Classify the compounds based on the image classification results.

Steps 2 to 7 of our method are described in greater detail in the following subsections.

4.1 Image Compression

MPEG is a state-of-the-art video compressor. It is appropriate for different applications because it supports spatial and temporal redundancy reduction resolution [30], is widely available, and has also highly optimized implementations [8]. According to our application, we need to find morphologically similar cells rather than identical ones. As MPEG is a lossy compressor and considering its advantages, we utilize MPEG encoder as our compressor. Using the MPEG encoder, we can set an image pixel search range according to cells' average size, which guarantees isolating a single cell from other cells in an image and acquire the cell's information of shape, size, brightness, etc. Thus, the information of two cell images is accurately compressed.

To measure the similarity of two images, the MPEG encoder creates a synthetic "video" of the two images. In this video, the first image is treated as a reference (R) frame, and the second image as a predicted (P) frame. Therefore, the MPEG encoder

can provide a total combined compressed size of the pair of images. If we use two images that are the same as input, this algorithm returns the compressed size of a single image; if we use two different images as input, the MPEG encoder will give the "between frames" (two images' inter-frame compression) compressed size plus the single frame (the first image's intra-frame compression) size. Several parameters must be set on the MPEG encoder. Taking [8] as a reference, a logarithmic P frame search algorithm is used for speed and consistency, the original images for intra-picture reference frames are used for the encoding step; and a bidirectional (B) frame quantization factor is ignored because no bidirectional frames are used in our method. Since the image size is 512*512 pixels, the integer search radius is set to 255. Because the average cell size is about 17 pixels, the quantization scales for R and P frames are set to 17 to maintain image compressibility and quality.

4.2 Image Information Distance Calculation

According to Eq. 6, our IDC method then uses the MPEG compressed image file size to compute the image information distance. Since the MPEG encoder returns the combined compressed size of a pair of images and the first image, the total combined compressed size of the pair of images must be represented by the compressed size of the two different images minus the size of the first image.

4.3 Control Images Statistics

Our IDC method uses control images from the wells of control treatments, as our training set. The whole batch of images has two known clusters, one labeled high control and one low control. It is assumed that the high/low control cluster contains n images indexed from 1 to n, and x and y are any two images within the same cluster. This method computes the mean value (μ) and standard deviation (σ) of the high/low control cluster, based on the information distance between any x and y.

4.4 Average Image Information Distance between Any Image and Control Images

The information distance between any image and any control image is calculated for image classification. The "any" image is indexed by x; HC means high control images, and LC means low control images. First, all the information distances between x and any control image ($HC(i_j)$ or $LC(i_j), (j \in [1, n])$) are calculated. To avoid the impact of extreme values, a geometric mean is chosen to represent the average value of information distances in this step. The average information distance between image x and all the high control(HC) images or low control(LC) images is represented by $geomean(ID(x, HC))$ or $geomean(ID(x, LC))$. The $geomean(ID(x, HC))$ means the geometric mean of the information distance sequence $ID(x, HC)$, and $geomean(ID(x, LC))$ means the sequence $ID(x, HC)$ where

$$\begin{aligned} ID(x, HC) &= ID(x, HC(i_1)), ID(x, HC(i_2)), \ldots, ID(x, HC(i_n)) \\ ID(x, LC) &= ID(x, LC(i_1)), ID(x, LC(i_2)), \ldots, ID(x, LC(i_n)) \end{aligned} \tag{7}$$

4.5 Image Classification

After the 4.3 Control Images Statistics step, there are two sets of mean values and standard deviations. One is μ_{HH} and σ_{HH}, which represent the mean value and standard variation of distance between any pair of high control images. The other is μ_{LL} and σ_{LL}, which show the mean value and variation of distance between any pair of low control images. Algorithm 1 classifies image x to either the high control or the low control cluster, according the values of $geomean(ID(x, HC))$, $geomean(ID(x, LC))$, μ_{HH}, σ_{HH}, μ_{LL}, σ_{LL}. As shown in Fig. 3, the distribution of distances between any two high/low control images is very similar to Gaussian distribution. Therefore, we use Gaussian distribution to model the distance distribution. If the average compressed size of image x and high control images is more similar to that of any two high control images, then image x will be classified as high control, and vice versa.

Algorithm 1. Image Classification

Require: $geomean(ID(x, HC))$,$geomean(ID(x, LC))$, $mpegSize(x, x)$
1: $X \leftarrow mpegSize(x, x)$
2: $XH \leftarrow geomean(ID(x, HC))$
3: $XL \leftarrow geomean(ID(x, LC))$
4: $XHNorm \leftarrow (XH - \mu_{HH})/\sigma_{HH}$
5: $XLNorm \leftarrow (XL - \mu_{LL})/\sigma_{LL}$
6: **if** $XHNorm \leqslant XLNorm$ **then**
7: x belongs to High Control
8: **else**
9: x belongs to Low Control
10: **end if**
11: **return** the cluster which x belongs to

It may be asked why no between-classes measurements are computed. Since our purpose is to classify test images into either high control or low control classes, we do not consider the distance between the two. That is, there is no need to compute σ_{HL} and μ_{HL}.

4.6 Compound Classification

Each compound is classified based on the information distances between its images. It is assumed that there are m high control compounds. Num_i is the number of labeled high control images contained in the i^{th} high control compound. Num_{tested} is the number of labeled high control images contained in a compound tested. Compounds tested are classified into active if the Num_{tested} is larger than $\min_i Num_i$.

All active compounds are identified after the above seven steps. Our method treats images as input, and gives the classification result directly. Based on an developed image similarity measure, our method utilizes the MPEG encoder with a fixed set of encoder-parameters to analyze GFP (cytoplasm) images and does not need Hoechst (nucleus) images. Therefore it avoids cytotoxic DNA staining, segmentation, and feature extraction steps.

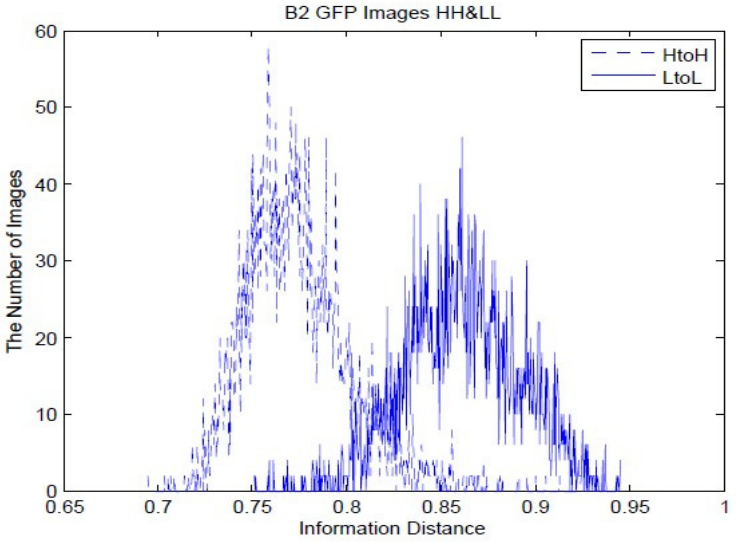


Fig. 3. Control image information distance distribution. They are the high and low control image information distance distributions of assay plate 2 of library plate 2. HtoH represents the information distance between any two high control images; and LtoL represents the information distance between any two low control images.

5 Results

In this section, we evaluate the effectiveness of the proposed framework for compound classification, using F_1 score as our performance measure. It is defined as in Eq. 8:

$$F_1 = \frac{2 * TP}{2 * TP \ + \ FP \ + \ FN} \tag{8}$$

where

$$TP \ : \ the\ number\ of\ true\ positive\ compounds;$$

$$FP \ : \ the\ number\ of\ false\ positive\ compounds;$$

$$FN \ : \ the\ number\ of\ false\ negative\ compounds.$$

We started by comparing the images of high and low control compounds. For short, we call these images high and low control images, respectively. Fig. 4 shows the information distance distribution of all six plates' high and low control images. The information distances between any pair of low control images were significantly larger than those between any pair of high control images.

Then we applied our IDC method to all images of the six plates' compounds. Using our method, six compounds in plate 1, five compounds in plate 2, six compounds in plate 3, three compounds in plate 4, seven compounds in plate 5, and six compounds in plate 6 were recognized as active compounds. All of the other compounds were treated

as inactive compounds. Fig. 5 indicates image samples taken randomly from both the active compounds (a) and inactive compounds (b). We referred to [29], and used an established commercial high content screening image analysis software – Acapella from PerkinElmer company [31] – to do the conventional analysis. Both methods in total identified 34 compounds as active; the other 446 compounds were inactive.

Those compounds flagged as active by either type of analysis were validated by further biological verification experiments. A full concentration (10 point dose) response curve for each compound was tested in 30 wells (10 concentrations in triplicate), and activity was considered "true" if the EC50 of the compound was less than 10μM. For the chosen compounds tested, the biological verification experiments follow the same process as the primary screening. Since the primary screening needs one week to culture cells, the verification experiments need another week, and the same monetary cost is involved as in the primary screening to test each chosen compound.

Table 1. Biological verification results on the conventional analysis and the IDC method

(a) Conventional Analysis Statistics

	Biological Experiment	
Conventional Analysis	$TruePositive$ = 29	$FalsePositive$= 5
	$FalseNegative$ = 3	$TrueNegative$= 9

(b) IDC Statistics

	Biological Experiment	
Information Distance	$TruePositive$ = 29	$FalsePositive$= 3
	$FalseNegative$ = 3	$TrueNegative$= 11

Due to the high related costs, the biological verification experiments did not test all compounds but focused mainly on those flagged as active by one or more computer-assisted methods to confirm whether those identified as positives were true positives. A few other biological verification experiments were run on compounds that were flagged as negatives to confirm whether they were true negatives.

In practice, the biological verification experiment picked 46 compounds from the total 480. These 46 compounds contain the total 34 active ones identified by the conventional analysis or the IDC method, or both. Table 2 shows the active compound identification results of the biological verification experiment, the conventional analysis and the IDC method. Table 1 shows the biological verification results for the conventional analysis and the IDC method using the 46 chosen compounds.

The F_1 scores of the conventional analysis and the IDC method are 0.88 and 0.91, respectively. Our IDC method achieves a 3% higher F_1 score than conventional analysis.

It may be perceived that the biological verification experiment may be biased, because it chose only 46 compounds that had already been shown to contain the 34 active compounds identified by union of the above two methods. However, this bias will not impact the comparison result, because it will only change those two negative values. While the absolute values of sensitivity and specificity will be changed as the total number of compounds confirmed is increased, the relative order of the sensitivity and specificity values of the two methods will not be changed.

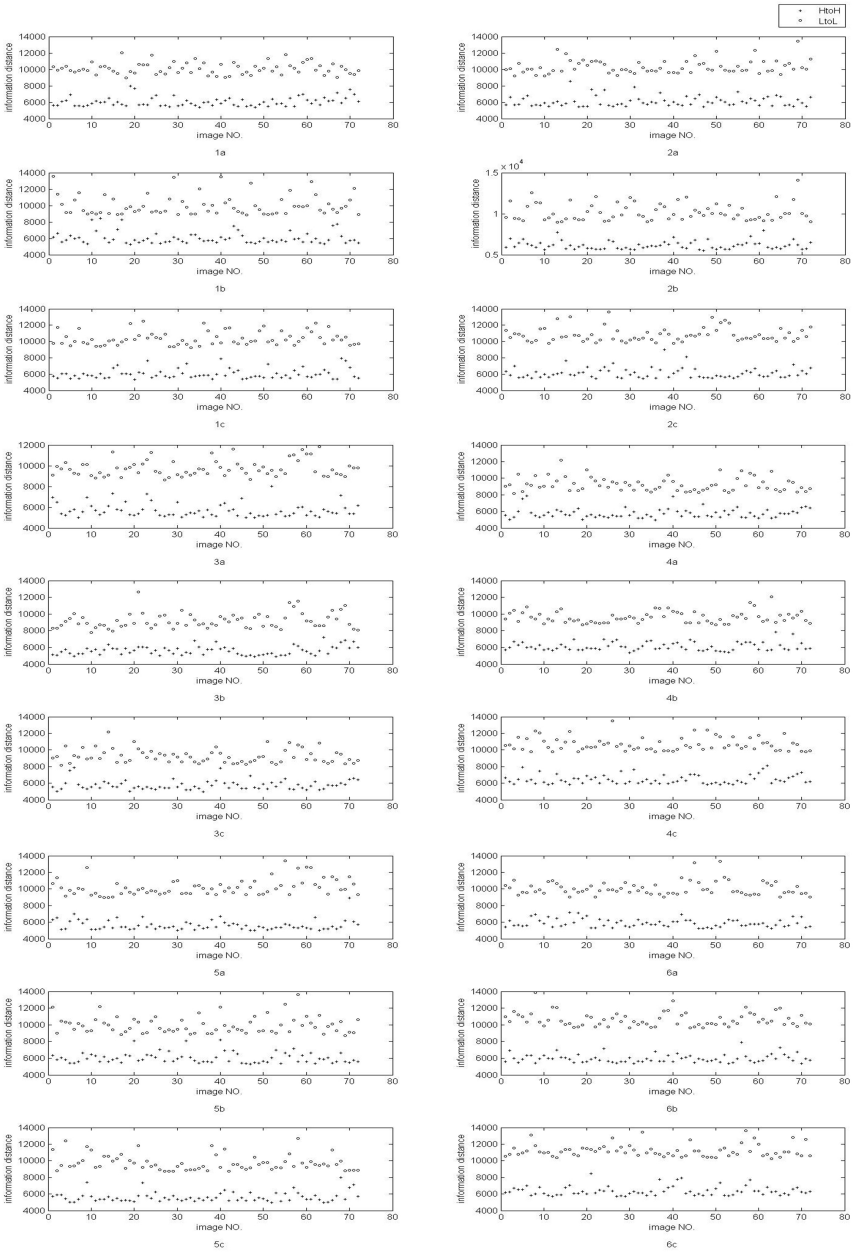
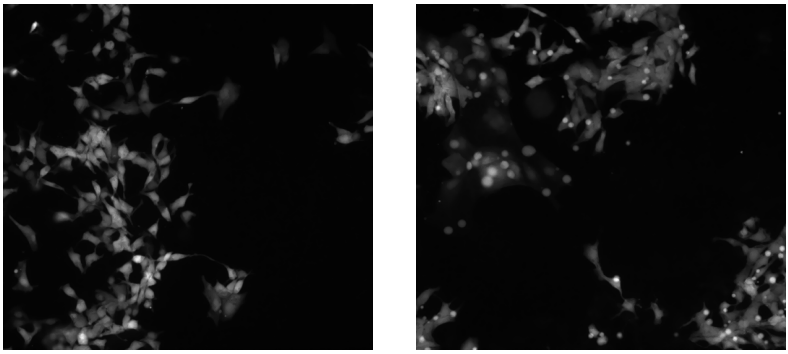


Fig. 4. The distribution of information distances between any pair of control images. The hollow circles represent the information distances between any pair of low control images, and the plus signs represent the information distances between any pair of high control images. The number of each subfigure index shows the ID number of the library plate, and the letter of that index represents one of the three different assay plates of a single library plate.



(a) Active Compound Sample (b) Inactive Compound Sample

Fig. 5. Image samples taken randomly from active compounds (a) and inactive compounds (b). Fig. 5(a) is from well E11 of plate 1, which was identified as a "true" active compound; Fig. 5(b) is from well E07 of plate 2, which was identified as a "true" inactive compound.

Table 2. Active compounds identification results of the biological verification experiment, the conventional analysis and the IDC method. This table shows the raw data of Table 1 in our paper. The checkmark indicates that the compound in the well is "active", while the backslash indicates that the compound in the well is "inactive".

No.	Plate No.	Well No.	BIO	CON	IDC
1	1	A06	\	✓	✓
2	1	E03	\	\	\
3	1	E07	✓	\	\
4	1	E08	\	✓	✓
5	1	E11	✓	✓	✓
6	1	F10	✓	✓	✓
7	1	G07	✓	✓	✓
8	1	G10	✓	✓	✓
9	1	H08	✓	\	\
10	2	A06	✓	✓	✓
11	2	A07	✓	✓	✓
12	2	A09	✓	✓	✓
13	2	B09	\	\	\
14	2	D06	✓	✓	✓
15	2	E07	\	✓	\
16	2	F07	\	✓	\
17	2	H08	✓	✓	✓
18	3	A04	✓	✓	✓
19	3	A10	✓	✓	✓
20	3	D02	\	\	\
21	3	D07	✓	✓	✓
22	3	G02	✓	\	\
23	3	G07	✓	✓	✓

No.	Plate No.	Well No.	BIO	CON	IDC
24	3	H02	\	\	\
25	3	H06	✓	✓	✓
26	3	H09	\	\	\
27	3	F04	\	\	\
28	4	B04	✓	✓	✓
29	4	C04	\	\	\
30	4	D09	\	\	\
31	4	E05	✓	✓	✓
32	4	F04	✓	✓	✓
33	5	B03	\	✓	✓
34	5	B04	✓	✓	✓
35	5	C11	✓	✓	✓
36	5	D06	\	\	\
37	5	E02	✓	✓	✓
38	5	F07	✓	✓	✓
39	5	H09	✓	✓	✓
40	5	H10	✓	✓	✓
41	6	A02	✓	✓	✓
42	6	A03	✓	✓	✓
43	6	B02	✓	✓	✓
44	6	B03	✓	✓	✓
45	6	G03	✓	✓	✓
46	6	G04	✓	✓	✓

6 Conclusion

Our methodology is based on information distance theory, and directly analyzes only cytoplasm images, with no need for DNA staining and image segmentation. Therefore, it avoids staining's cytotoxic/cytostatic problems and segmentation errors, and allows kinetic studies. Since it compresses images directly, there is no need to extract any features. In particular, it does not require a human intervention step, as is required in the conventional analysis, to obtain image information such as cell count and brightness.

Compared with the conventional analysis method, the IDC method achieves a higher F1 score. It is simpler to use, and acquires better results.

Acknowledgements. This work was funded by the Chinese Scholarship Council and was supported (in part) by an Ontario Ministry of Economic Development and Innovation (MEDI) grant.

References

1. Jaenisch, R., Young, R.: Stem Cells, the Molecular Circuitry of Pluripotency and Nuclear Reprogramming. Cell 132, 567–582 (2008)
2. Ding, S., Wu, T.Y.H., Brinker, A., Peters, E.C., Hur, W., Gray, N.S., Schultz, P.G.: Synthetic small molecules that control stem cell fate. PNAS 100, 7632–7637 (2003)
3. Ljosa, V., Carpenter, A.E.: Introduction to the Quantitative Analysis of Two-Dimensional Fluorescence Microscopy Images for Cell-Based Screening. Plos Computational Biology 5, 1–10 (2009)
4. Li, M., Vitanyi, P.: An introduction to Komogorov complexity and its applications. Springer, New York (1997)
5. Bennett, C.H., Gacs, P., Li, M., Vitanyi, P., Zurek, W.: Information Distance. IEEE Trans. Inform. Theory 44, 1407–1423 (1993)
6. Arbuchle, T., Balaban, A., Peters, D.K., Lawford, M.: Software documents: comparison and measurement. In: Proceeding 18th International Conference on Software Engineering & Knowledge Engineering, Boston, USA, pp. 740–748 (2007)
7. Anë, C., Sanderson, M.J.: Missing the Forest for the Trees: Phylogenetic Compression and Its Implications for Inferring Complex Evolutionary Histories. J. Sys. Biol. 54, 146–157 (2005)
8. Campana, B.J.L., Keogh, E.J.: A Compression-Based Distance Measure for Texture. J. Statistical Analysis and Data Mining 3, 381–398 (2010)
9. Cerra, D., Mallet, A., Gueguen, L., Datcu, M.: Algorithmic Information Theory-Based Analysis of Earth Observation Images: An Assessment. J. IEEE Geoscience and Remote Sensing Letters 7, 8–12 (2010)
10. Chen, X., Francia, B., Li, M., Mckinnon, B., Seker, A.: Shared information and program plagiarism detection. IEEE Trans. Info. Theory 50, 1545–1550 (2004)
11. Cilibrasi, R., Vitänyi, P.M.B., de Wolf, R.: Algorithmic clustring of music based on string compression. J. Comput. Music 28, 49–67 (2004)
12. Cilibrasi, R., Vitänyi, P.M.B.: Clustering by compression. IEEE Trans. Knowledge & Data Engineering 19, 370–383 (2007)
13. Cohen, A.R., Bjornsson, C.S., Temple, S., Banker, G., Roysam, B.: Automatic Summarization of Changes in Biological Image Sequences Using Algorithmic Information Theory. IEEE Trans. Pattern Analysis & Machine Intelligence 31, 1386–1403 (2009)

14. Cuturi, M., Vert, J.P.: The context-tree kernel for strings. Neural Networks 18, 1111–1123 (2005)
15. Benedetto, D., Caglioti, E., Loreto, V.: Language trees and zipping. Phys. Rev. Lett. 88, 48702 (2002)
16. Kocsor, A., Kertesz, F.A., Kajan, L., Pongor, S.: Application of compression-based distance measures to protein sequence classification: a methodology study. Bioinformatics 22, 407–412 (2006)
17. Kirk, S.R., Jenkins, S.: Information theory-based software metrics and obfuscation. J. Systems and Software 72, 179–186 (2004)
18. Krasnogor, N., Pelta, D.A.: Measuring the similarity of protein structures by means of the universal similarity metric. Bioinformatics 20, 1015–1021 (2004)
19. Kraskov, A., Stögbauer, H., Andrzejak, R.G., Grassberger, P.: Hierarchical clustering using mutual information. Europhys. Lett. 70, 278–284 (2005)
20. Li, M., Badger, J.H., Chen, X., Kwong, S., Kearney, P., Zhang, H.Y.: An information-based sequence distance and its application to whole mitochondrial genome phylogeny. Bioinformatics 17, 149–154 (2001)
21. Nikvand, N., Wang, Z.: Generic Image Similarity Based on Kolmogorov Complexity. In: 17th IEEE International Conference on Image Processing, pp. 309–312. IEEE Press, Hong Kong (2010)
22. Otu, H.H., Sayood, K.: A new sequence distance measure for phy6logenetic tree construction. Bioinformatics 19, 2122–2130 (2003)
23. Pao, H.K., Case, J.: Computing entropy for ortholog detection. In: International Conference on Computational Intelligence, Istanbul, Turkey, pp. 89–92 (2004)
24. Parry, D.: Use of Kolmogorov distance identification of web page authorship, topic and domain. In: Workshop on Open Source Web Information Retrieval (2005)
25. Perkiö, J., Hyvärinen, A.: Modelling Image Complexity by Independent Component Analysis, with Application to Content-Based Image Retrieval. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) ICANN 2009, Part II. LNCS, vol. 5769, pp. 704–714. Springer, Heidelberg (2009)
26. Santos, C.C., Bernardes, J., Vitányi, P.M.B., Antunes, L.: Clustering fetal heart rate tracings by compression. In: Proceeding 19th IEEE Internation Symposium Computer-Based Medical Systems, Salt Lake City, pp. 22–23 (2006)
27. Zhang, X., Hao, Y., Zhu, X.Y., Li, M.: Information Distance from a Question to an Answer. In: KDD, San Jose, pp. 12–15 (2007)
28. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removel algorithms. Physica D 60, 259–268 (1992)
29. Panchal, R.G., Kota, K.P., Spurgers, K.B., Ruthel, G., Tran, J.P., Boltz, R.C., Bavari, S.: Development of High-Content Imaging Assays for Lethal Viral Pathogens. J. Biomol. Screen 15, 755–765 (2010)
30. Gall, D.L.: Mpeg: a video compression standard for multimedia application. Commun. ACM 34, 46–58 (1991)
31. http://www.perkinelmer.com/pages/020/cellularimaging/products/acapella.xhtml

Distance Metric Learning Revisited

Qiong Cao[1], Yiming Ying[1], and Peng Li[2]

[1] College of Engineering, Mathematics and Physical Sciences,
University of Exeter, Harrison Building, Exeter, EX4 4QF, UK
{qc218,y.ying}@exeter.ac.uk
[2] Department of Engineering Mathematics,
University of Bristol, Bristol, BS8 1UB, UK
lipeng@ieee.org

Abstract. The success of many machine learning algorithms (e.g. the nearest neighborhood classification and k-means clustering) depends on the representation of the data as elements in a metric space. Learning an appropriate distance metric from data is usually superior to the default Euclidean distance. In this paper, we revisit the original model proposed by Xing et al. [25] and propose a general formulation of learning a Mahalanobis distance from data. We prove that this novel formulation is equivalent to a convex optimization problem over the spectrahedron. Then, a gradient-based optimization algorithm is proposed to obtain the optimal solution which only needs the computation of the largest eigenvalue of a matrix per iteration. Finally, experiments on various UCI datasets and a benchmark face verification dataset called Labeled Faces in the Wild (LFW) demonstrate that the proposed method compares competitively to those state-of-the-art methods.

Keywords: Metric learning, convex optimization, Frank-Wolfe algorithm, face verification.

1 Introduction

Many machine learning algorithms critically depend on the quality of the chosen distance metric. For instance, k-nearest neighbor classification needs the identification of nearest neighbors and k-means clustering depends on the distance measurements for clustering. The default distance is the Euclidean distance, which, however, does not reflect the given data representation. Recent advances in metric learning [1,2,4,6,19,20,22,23,25,27] make it possible to learn an effective distance metric which is more suitable for a given learning problem. These methods have demonstrated the successful applications of metric learning to various real-world problems including information retrieval and face verification.

Given some partial information of constraints, the goal of metric learning is to learn a distance metric which reports small distances for *similar* examples and large distances for *dissimilar* examples. The partial information can be presented in the form of constraints such as similarity or dissimilarity between a pair of examples. These constraints can be collected either from the label information

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 283–298, 2012.

in supervised classification or the side information in semi-supervised clustering such as must-links and cannot-links. Most of metric learning methods focus on learning a Mahalanobis metric defined by $d_M(x_i, x_j) = \sqrt{(x_i - x_j)^\top M(x_i - x_j)}$ where M is a positive semi-definite (p.s.d.) matrix. Many metric learning methods for learning Mahalanobis distances are therefore formulated as semi-definite programs [21].

Depending on the generation of constraints information, metric learning can be supervised or unsupervised. Unsupervised metric learning is closely related to dimension reduction. To see this, observe that any positive semi-definite M can be rewritten as $A^\top A$, and hence, $d_M(x_i, x_j) = \sqrt{(x_i - x_j)^\top M(x_i - x_j)} = \|A(x_i - x_j)\|$. This simple observation implies that learning an appropriate M is equivalent to learning an appropriate projection map A. From this perspective, dimension reduction methods (e.g. [3,16,17]) can be regarded as unsupervised metric learning. In supervised metric learning, the available labels can be used to create the information of constraints. Supervised metric learning can be further divided into two categories: the global method and the local method. The global methods learn the distance metric which satisfies all the pairwise constraints simultaneously. The original model proposed by Xing et al. [25] is a global method which used all the similar pairs (same labels) and dissimilar pairs (distinct labels). Local methods only use local pairwise constraints which usually outperform the global ones as observed in many previous studies. This is particularly reasonable in the case of learning a metric for the kNN classifiers since kNN classifiers are influenced mostly by the data items that are close to the test/query examples. Since we are mainly concerned with metric learning for kNN classifier, the pairwise constraints are generated locally, that is, the similar/dissimilar pairs are k-nearest neighbors. The details can be found in the experimental section.

In this paper, we revisit the original model proposed by Xing et al. [25], where the authors proposed to learn a metric by maximizing the distance between dissimilar samples whilst keeping the distance between similar points upper-bounded. However, the projection gradient method employed there usually takes a large number of iterations to become convergent, and also it needs the full eigen-decomposition per iteration. The first contribution of this paper is to extend the methods in [25,28] and propose a general formulation for metric learning. We prove the convexity of this general formulation and illustrate it with various examples. Our second contribution is to show, by exploring its special structures, that the proposed formulation is further equivalent to a convex optimization over the spectrahedron. This equivalent formulation enables us to directly employ the Frank-Wolfe algorithm [5] to obtain the optimal solution. In contrast to the algorithm in [25], our proposed algorithm only needs to compute the largest eigenvalue of a matrix per iteration and is guaranteed to converge with a time complexity $\mathcal{O}(1/t)$ where t is the iteration number.

The paper is organized as follows. The next section presents the proposed model and proves its convexity. Section 3 establishes its equivalent formulation from which an efficient algorithm is proposed. In Section 4, we review and discuss

some related work on metric learning. Section 5 reports experimental results on UCI datasets and a benchmark face verification dataset called Labeled Faces in the Wild (LFW). The last section concludes the paper.

2 Convex Metric Learning Model

We begin by introducing some useful notations. For any $n \in \mathbb{N}$, denote $\mathbb{N}_n = \{1, 2, \ldots, n\}$. The space of symmetric $d \times d$ matrices is denoted by $\mathbb{S}^d$ and $\mathbb{S}^d_+$ denotes the cone of positive semi-definite matrices. For any $X, Y \in \mathbb{R}^{d\times n}$, the inner product in $\mathbb{S}^d$ is denoted by $\langle X, Y\rangle := \mathbf{Tr}(X^\top Y)$ where $\mathbf{Tr}(\cdot)$ is the trace of a matrix.

For simplicity, we focus on learning a distance metric for kNN classification, although the proposed methods below can easily be adapted to metric learning for k-means clustering. Now we denote the training data by $\mathbf{z} := \{(x_i, y_i) : i \in \mathbb{N}_n\}$ with input $x_i = (x_i^1, x_i^2, \ldots, x_i^d) \in \mathbb{R}^d$, class label y_i (not necessary binary). Later on, we use the convention $X_{ij} = (x_i - x_j)(x_i - x_j)^\top$ and let $\mathcal{S}$ index the similarity pairs, $\mathcal{D}$ index the dissimilarity pairs. For instance, $\tau = (i, j) \in \mathcal{S}$ means that (x_i, x_j) is a similar pair and rewrite X_{ij} as X_τ. One can follow the mechanism in [22] to extract local information of similarity or dissimilarity for kNN classification; see the experimental section for more details.

Given a set of similar samples and a set of dissimilar samples, we aim to find a good distance matrix M such that the distance between the dissimilar pair is large while keeping the distance between the similar pairs small. There are many formulations to achieve this goal. In particular, the following formulation was proposed in [25]:

$$\begin{array}{cc} \max_{M\in\mathbb{S}^d_+} & \sum_{(i,j)\in\mathcal{D}} d_M(x_i, x_j) \\ \text{s.t.} & \sum_{(i,j)\in\mathcal{S}} [d_M(x_i, x_j)]^2 \le 1. \end{array} \tag{1}$$

An iterative projection method was employed to solve the above problem. However, the algorithm generally takes a long time to converge and it needs the computation of the full eigen-decomposition of a matrix per iteration.

In this paper, we propose a more general formulation:

$$\begin{array}{cc} \max_{M\in\mathbb{S}^d_+} & \left[\sum_{(i,j)\in\mathcal{D}} [d_M(x_i, x_j)]^{2p}/D\right]^{\frac{1}{p}} \\ \text{s.t.} & \sum_{(i,j)\in\mathcal{S}} [d_M(x_i, x_j)]^2 \le 1, \end{array} \tag{2}$$

where $p \in (-\infty, \infty)$ and D is the number of dissimilarity pairs. We refer to the above formulation as $\mathbf{DML}_p$. The above formulation is well defined even for the limiting case $p = 0$ as discussed in the examples below.

- $\mathbf{p = 1/2}$: In this case, problem (2) can be written as

$$\begin{array}{cc} \max_{M\in\mathbb{S}^d_+} & \left[\sum_{(i,j)\in\mathcal{D}} d_M(x_i, x_j)/D\right]^2 \\ \text{s.t.} & \sum_{(i,j)\in\mathcal{S}} [d_M(x_i, x_j)]^2 \le 1, \end{array} \tag{3}$$

  which is equivalent to formulation (1) proposed in [25].

– $\mathbf{p} \to -\infty$: Observe, for any positive sequence $\{\alpha_i > 0 : i \in \mathbb{N}_n\}$, that

$$\lim_{p\to-\infty} \big(\sum_{i\in\mathbb{N}_n} a_i^p/n\big)^{\frac{1}{p}} = \min_{i\in\mathbb{N}_n} a_i.$$

Hence, in the limiting case $p \to -\infty$, problem (2) is reduced to the metric learning model called DML-eig [28]:

$$\begin{array}{cc} \max_{M\in\mathbb{S}_+^d} & \min_{(i,j)\in\mathcal{D}}[d_M(x_i,x_j)]^2 \\ \text{s.t.} & \sum_{(i,j)\in\mathcal{S}}[d_M(x_i,x_j)]^2 \le 1. \end{array} \tag{4}$$

– $\mathbf{p} \to \mathbf{0}$: Note, for any sequence $\{\alpha_i > 0 : i \in \mathbb{N}_n\}$, that

$$\lim_{p\to 0}\big[\sum_{i\in\mathbb{N}_n} a_i^p/n\big]^{\frac{1}{p}} = \prod_{i=1}^{n} \alpha_i^{\frac{1}{n}}.$$

Hence, in the limiting case $p \to 0$, problem (2) becomes

$$\begin{array}{cc} \max_{M\in\mathbb{S}_+^d} & \prod_{(i,j)\in\mathcal{D}}[d_M(x_i,x_j)]^{\frac{2}{D}} \\ \text{s.t.} & \sum_{(i,j)\in\mathcal{S}}[d_M(x_i,x_j)]^2 \le 1, \end{array}$$

where D is the number of dissimilar pairs in the set $\mathcal{D}$.

The following theorem investigates the convexity/concavity of the objective function in problem (2).

Theorem 1. *Let function $\mathcal{L} : \mathbb{S}_+^d \to \mathbb{R}$ be the objective function of DML$_p$, i.e., for any $M \in \mathbb{S}_+^d$, $\mathcal{L}(M) = \big[\sum_{(i,j)\in\mathcal{D}}\langle X_{ij}, M\rangle^p/D\big]^{\frac{1}{p}}$ for $p \neq 0$, and $\mathcal{L}(M) = \prod_{(i,j)\in\mathcal{D}}[d_M(x_i,x_j)]^{\frac{2}{D}}$ for $p = 0$. Then, we have that $\mathcal{L}(\cdot)$ is concave for $p < 1$ and otherwise convex.*

Proof. First we prove the concavity of $\mathcal{L}(\cdot)$ when $p < 1$ and $p \neq 0$. It suffices to prove, for any $n \in \mathbb{N}$ and for any $\{\mathbf{a} = (a_1, a_2, \ldots, a_n) : a_i > 0, \quad i \in \mathbb{N}_n\}$, that function $(\sum_{j\in N_n} a_j^p)^{1/p}$ is concave w.r.t. variable $\mathbf{a}$. To this end, let f be a function defined, for any $x > 0$ and $y > 0$, by $f(x, y) = -x^{1-p}y^p/p$. We can easily prove that f is jointly convex w.r.t. (x, y), since its Hessian matrix

$$(1-p)\begin{pmatrix} x^{-p-1}y^p & -x^{-p}y^{p-1} \\ -x^{-p}y^{p-1} & x^{1-p}y^{p-2} \end{pmatrix} \in \mathbb{S}_+^d.$$

Consequently, for any $i \in \mathbb{N}_n$, $-x^{1-p}a_i^p/p$ is jointly convex, which implies that its summation $\sum_{i\in\mathbb{N}_n} -x^{1-p}a_i^p/p = -x^{1-p}(\sum_{i\in\mathbb{N}_n} a_i^p)/p$ is jointly convex. Hence, the function defined by $E(x, \mathbf{a}) = (1-p)x/p - x^{1-p}(\sum_{i\in\mathbb{N}_n} a_i^p)/p$ is also jointly convex w.r.t. $(x, \mathbf{a})$. Clearly,

$$-\big(\sum_{j\in N_n} a_j^p\big)^{1/p} = \min\{E(x, \mathbf{a}) : x \ge 0\}. \tag{5}$$

Recalling that the partial minimum of a jointly convex function is convex [9, Sec.IV.2.4], we obtain the concavity of $(\sum_{j\in N_n} a_j^p)^{1/p}$ when $p < 1$ and $p \neq 0$. The concavity of $\mathcal{L}$ for $p = 0$ follows from the fact that the limit function of a sequence of concave functions is concave.

The convexity of $\mathcal{L}$ for $p \geq 1$ can be proved similarly by observing that $E(x, \mathbf{a})$ is jointly concave if $p \geq 1$. Consequently, equation (5) should be replaced by $(\sum_{j\in N_n} a_j^p)^{1/p} = \min\{-E(x, \mathbf{a}) : x \geq 0\}$. This completes the proof of the theorem.

We conclude this section with two remarks. Firstly, we exclude the extreme case $p = 1$ since, in this case, the optimal solution of DML$_p$ will be always a rank-one matrix (i.e. the data is projected to the line), as argued in [25]. Secondly, when $p \in (1, \infty)$, by Theorem 1 we know that formulation (2) is indeed a problem of *maximizing a convex function*, which is a challenging task to get a global solution. In this paper we will only consider the case $p \in (-\infty, 1)$ which guarantees that formulation (2) is a convex optimization problem.

3 Equivalent Formulation and Optimization

We turn our attention to an equivalent formulation of problem (2), which is critical to designing its efficient algorithms. For notational simplicity, denote the *spectrahedron* by $\mathcal{P} = \{M \in \mathbb{S}_+^d : \mathbf{Tr}(M) = 1\}$ and let $X_\mathcal{S} = \sum_{(i,j)\in\mathcal{S}} X_{ij}$. Then, DML$_p$ (i.e. formulation (2)) can be rewritten as the following problem:

$$\begin{array}{ll} \max_{M\in\mathbb{S}_+^d} & \left[\sum_{\tau\in\mathcal{D}} \langle X_\tau, M\rangle^p / D\right]^{\frac{1}{p}} \\ \text{s.t.} & \langle X_\mathcal{S} + \delta \mathbf{I}_d, M\rangle \leq 1. \end{array} \tag{6}$$

Without loss of generality, we assume that $X_\mathcal{S}$ is invertible throughout the paper. This can be achieved by adding a small ridge term, i.e. $X_\mathcal{S} \longleftarrow X_\mathcal{S} + \delta\, \mathbf{I}_d$ where $\mathbf{I}_d$ is the identity matrix and $\delta > 0$ is a small ridge constant. In this case, we can apply the Cholesky decomposition to get that $X_\mathcal{S} = LL^\top$, where L is a lower triangular matrix with strictly positive diagonal entries.

Equipped with the above preparations, we are now ready to show that problem (2) is equivalent to an optimization problem over the spectrahedron $\mathcal{P} = \{M \in \mathbb{S}_+^d : \mathbf{Tr}(M) = 1\}$. Similar ideas have been used in [28].

Theorem 2. *For any $\tau = (i, j) \in \mathcal{D}$, let $\widetilde{X}_\tau = L^{-1}(x_i - x_j)(L^{-1}(x_i - x_j))^\top$. Then, problem (2) is equivalent to*

$$\max_{S\in\mathcal{P}} \Big[\sum_{\tau\in\mathcal{D}} \langle \widetilde{X}_\tau, S\rangle^p\Big]^{\frac{1}{p}}, \tag{7}$$

Proof. Let M^* be an optimal solution of problem (2) and $\widetilde{M}^* = \frac{M^*}{\langle X_\mathcal{S}, M^*\rangle}$. Then, $\langle X_\mathcal{S}, \widetilde{M}^*\rangle = 1$ and $\left[\sum_{\tau\in\mathcal{D}} \frac{\langle X_\tau, \widetilde{M}^*\rangle^p}{D}\right]^{\frac{1}{p}} = \left[\sum_{\tau\in\mathcal{D}} \frac{\langle X_\tau, M^*\rangle^p}{D}\right]^{\frac{1}{p}} / \langle X_\mathcal{S}, M^*\rangle \geq$

Table 1. Pseudo-code of the Frank-Wolfe algorithm to solve DML_p where f denotes the objective function of formulation (7)

Input:
· parameter $p \in (-\infty, 1)$
· tolerance value *tol* (e.g. 10^{-5})
· step sizes $\{\alpha_t = 2/(t+1) : t \in \mathbb{N}\}$
Initialization: $S_1 \in \mathbb{S}_+^d$ with $\mathbf{Tr}(S_1) = 1$
for $t = 1, 2, 3, \ldots$ **do**
· $Z_t = \arg\max \left\{ \langle Z, \nabla f(S_t) \rangle : Z \in \mathbb{S}_+^d,\ \mathbf{Tr}(Z) = 1 \right\}$ i.e. $Z_t = vv^\top$
where v is the maximal eigenvector of matrix $\nabla f(S_t)$
· $S_{t+1} = (1 - \alpha_t) S_t + \alpha_t Z_t$
· if $|f(S_{t+1}) - f(S_t)| < tol$ then **break**
Output: $d \times d$ matrix $S_t \in \mathbb{S}_+^d$

$\left[\sum_{\tau \in \mathcal{D}} \frac{\langle X_\tau, M^* \rangle^p}{D}\right]^{\frac{1}{p}}$ since $\langle X_\mathcal{S}, M^* \rangle \leq 1$. This implies that $\widetilde{M}^*$ is also an optimal solution. Consequently, problem (2) is equivalent to, up to a scaling constant,

$$\begin{array}{ll} \max_{M \in \mathbb{S}_+^d} & \left[\sum_{(i,j) \in \mathcal{D}} \langle X_\tau, M \rangle^p / D\right]^{\frac{1}{p}} \\ \text{s.t.} & \langle X_\mathcal{S}, M \rangle = 1. \end{array} \tag{8}$$

Recall that $X_\mathcal{S} = LL^\top$ by Cholesky decomposition. Now the desired equivalence between (2) and (7) follows from changing variable $S = L^\top M L$ in (8). This completes the proof of the theorem.

By Theorem 2, the original metric learning problem (2) is reduced to a maximization problem on the spectrahedron. Therefore, we can apply the Frank-Wolfe (FW) algorithm [5,8] to obtain the optimal solution: the pseudo-code of the algorithm is given in Table 1 where f denotes the objective function of formulation (7). We conclude this section with a final remark. The objective function $\left[\sum_{\tau \in \mathcal{D}} \langle \widetilde{X}_\tau, S \rangle^p\right]^{\frac{1}{p}}$ in formulation (7) is not smooth since p can be negative. In order to avoid the numerical instability, we can add a small positive number inside so that it becomes a smooth function, i.e. $\left[\sum_{\tau \in \mathcal{D}} (\langle \widetilde{X}_\tau, S \rangle)^p\right]^{\frac{1}{p}}$ is replaced by $\left[\sum_{\tau \in \mathcal{D}} (\langle \widetilde{X}_\tau, S \rangle + \varepsilon)^p\right]^{\frac{1}{p}}$ where ε is a small positive number (e.g. $\varepsilon = 10^{-8}$). If the objective function has a Lipschitz-continuous gradient, then, by choosing $\alpha_t = \frac{2}{t+1}$, the FW algorithm is guaranteed to converge with a time complexity $\mathcal{O}(1/t)$. One can refer to [8,27] for a detailed proof.

4 Related Work

In recent years, distance metric learning has received a lot of attention in machine learning, see e.g. [1,2,4,6,15,19,20,22,25,27] and the references therein. It will be a difficult task to give a comprehensive review on related work. Below we only

briefly discuss some methods which are closely related to our work. We refer the readers to [26] for more related work on metric learning.

Xing et al. [25] presented metric-learning formulation (1) for k-means clustering. The method aims to maximize the distances between dissimilar samples subject to the constraint that distances between similar samples are upper-bounded. Ying et al. [28] proposed to maximize the minimal distance between dissimilar pairs while maintaining an upper bound for the distances between similar pairs. The proposed method (4) was shown to be equivalent to an eigenvalue optimization, which was solved by the Frank-Wolfe algorithm after smoothing the objective function. Our method DML_p is mainly motivated by the above two methods and provides a more general framework by recovering [25,28] as special cases. In contrast to the alternating projection method [25], we show that DML_p is reduced to a convex optimization problem over the spectrahedron. This new optimization formulation enables the direct application of the Frank-Wolfe algorithm which only needs the computation of the largest eigenvector of a matrix per iteration.

Weinberger et al. [22] developed the method called LMNN to learn a Mahalanobis distance metric in kNN classification settings. LMNN, as one of the state-of-the-art metric learning methods, aims to enforce k-nearest neighbors always belonging to the same class while examples from different classes being separated by a large margin. LMNN is a local method as it only used triplets from k-nearest neighbors. Similar to LMNN, our method focuses on similar pairs and dissimilar pairs generated from k-nearest neighbors. Davis et al. [4] proposed an information theoretic approach (ITML) to learning a Mahalanobis distance function by minimizing the Kullbach-Leibler divergence between two multivariate Gaussians subject to pairwise constraints.

Shen et al. [19] recently employed the exponential loss for metric learning named as BoostMetric and a boosting-based algorithm was developed. The rationale behind this algorithm is that each p.s.d. matrix can be decomposed into a linear positive combination of trace-one and rank-one matrices. This algorithm is very similar to the Frank-Wolfe algorithm employed for DML_p since both of them iteratively find a linear combination of rank-one matrices to approximate the desired solution. However, the method is a general column-generation algorithm and its convergence rate is not clear. The Frank-Wolfe algorithm for DML_p is theoretically guaranteed to have a convergence rate $\mathcal{O}(1/t)$ and it is relatively easy to be implemented by using just a few lines of MATLAB codes.

Guillaumin et al. [7] presented a metric learning model based on a logistic regression loss function called LDML. The method aims to learn robust distance measures for face identification using a logistic discriminant. In order to reduce the computational time, the authors proposed to remove the positive semi-definiteness constraint on the distance matrix. This would only lead to a sub-optimal solution.

Table 2. Description of datasets used in the experiments: n and d respectively denote the number of samples and attributes (feature elements) of the data; T is the number of triplets and D is the number of dissimilar pairs

Data	No.	n	d	class	T	D
Balance	1	625	4	3	3951	1317
Breast-Cancer	2	569	30	2	3591	1197
Diabetes	3	768	8	2	4842	1614
Image	4	2310	19	2	14553	4851
Iris	5	150	4	3	954	315
Waveform	6	5000	21	3	31509	10503
Wine	7	178	13	3	1134	378

5 Experiments

In this section, we compare the empirical performance of our proposed method DML_p with six other methods: the method proposed in [25] denoted by *Xing*, *LMNN* [22], *ITML* [4], *BoostMetric* [19], *DML-eig* [28] and the baseline algorithm using the standard Euclidean distance denoted by *Euclidean*. The model parameters in ITML, LMNN, BoostMetric and DML_p are tuned via three-fold cross validation. In addition, the maximum iteration number for DML_p is 1000 and the algorithm is terminated when the relative change of the objective function value is less than 10^{-5}.

We first run the experiments on UCI datasets to compare the kNN classification performance ($k = 3$) of different metric learning methods, where the kNN classifier is constructed using the Mahalanobis distance learned by metric learning methods. Then, we investigate the application of our method to the problem of face verification. In particular, we evaluate our new metric learning method using a large scale face database called Labeled Faces in the Wild (LFW) [10]. The LFW dataset is very challenging and difficult due to face variations in scale, pose, lighting, background, expression, hairstyle, and glasses, as the faces are detected in images in the wild, taken from Yahoo! News. Recently it has become a benchmark to test new face verification algorithms [10,24,7,18].

5.1 Convergence and Generalization on UCI Datasets

To investigate the convergence and generalization of DML_p , we run experiments on seven UCI datasets: i.e. 1) Balance; 2) Breast-Cancer; 3) Diabetes; 4) Image segmentation; 5) Iris; 6) Waveform; 7) Wine. The statistics of the datasets are summarized in Table 2. All the experimental results are obtained by averaging over 10 runs and, for each run, the data is randomly split into 70% for training and 30% for testing. To generate relative constraints and pairwise constraints, we adopt a similar mechanism in [22]. More specifically, for each training point x_i, k nearest neighbors that have the same labels as y_i (targets) as well as k nearest neighbors that have different labels from y_i (imposers) are found. According

to x_i and its corresponding targets and imposers, we then construct the set of similar pairs $\mathcal{S}$, the set of dissimilar pairs $\mathcal{D}$ and the set of relative constraints in the form of triplets denoted by $\mathcal{T}$ required by LMNN and BoostMetric. As mentioned above, the original formulation in [25] used all pairwise constraints. For fairness of comparison, all methods including *Xing* used the same set of similar/dissimilar pairs generated locally as above.

Firstly, we study the convergence of algorithm DML$_p$ with varying values of p. In Figure 1, we plot the objective function value of DML$_p$ versus the number of iteration on Balance (subfigure (a)); Iris (subfigure (b)); Diabetes (subfigure (c)); and Image (subfigure (d)). We can see from Figure 1 that the algorithm converges quickly. The smaller the value of p is and the more iterations algorithm DML$_p$ needs.

Secondly, we investigate the performance of DML$_p$ against different values of p. Figure 2 depicts the test error of DML$_p$ versus the value of p on Balance (subfigure (a)); Iris (subfigure (b)); Diabetes (subfigure (c)); and Image (subfigure (d)). We can observe from Figure 2 that the test error varies on different values of p and the best performance of DML$_p$ is superior to those of DML-eig [28] and Xing [25] which are the special cases of DML$_p$ with $p \to -\infty$ and $p = 1/2$ respectively. This observation validates the value of the general formulation DML$_p$ and suggests the importance of choosing an appropriate value of p. In the following experiments, we will tune the value of p by three cross-validation.

Finally, we study the generalization performance of kNN classifiers where the distance metric to measure nearest neighbors is learned by metric learning methods. To this end, we compare DML$_p$ with other metric learning methods including Xing [25], LMNN [22,23], ITML [4] and BoostMetric [19] as mentioned above. Figure 3 depicts the performance of different methods. It shows that almost all metric learning methods improve kNN classification using Euclidean distance on most datasets. Our proposed method DML$_p$ delivers competitive performance with other state-of-the-art algorithms such as ITML, LMNN and BoostMetric. Indeed, DML$_p$ outperforms other methods on 4 out of 7 datasets and shows competitive performance against the best one on the rest 3 datasets. From Figure 3, it is reasonable to see that the test errors of DML$_{1/2}$ are consistent with those of Xing since they are essentially the same model implemented by different algorithms. The only exception is the performance on Waveform dataset: the test error of Xing is much worse than DML$_{1/2}$. The reason could be that the alternating projection method proposed in [25] does not converge in a reasonable time due to the relatively large number of samples in Waveform dataset.

5.2 Application to Face Verification

The task of face verification is to determine whether two face images are from the same identity or not. Metric learning provides a very natural solution by comparing the image pairs based on the metric learnt from the face data. In this experiment, we investigate the performance of DML$_p$ on the LFW dataset [10] – a benchmark dataset for face verification. It contains a total of 13233 labeled

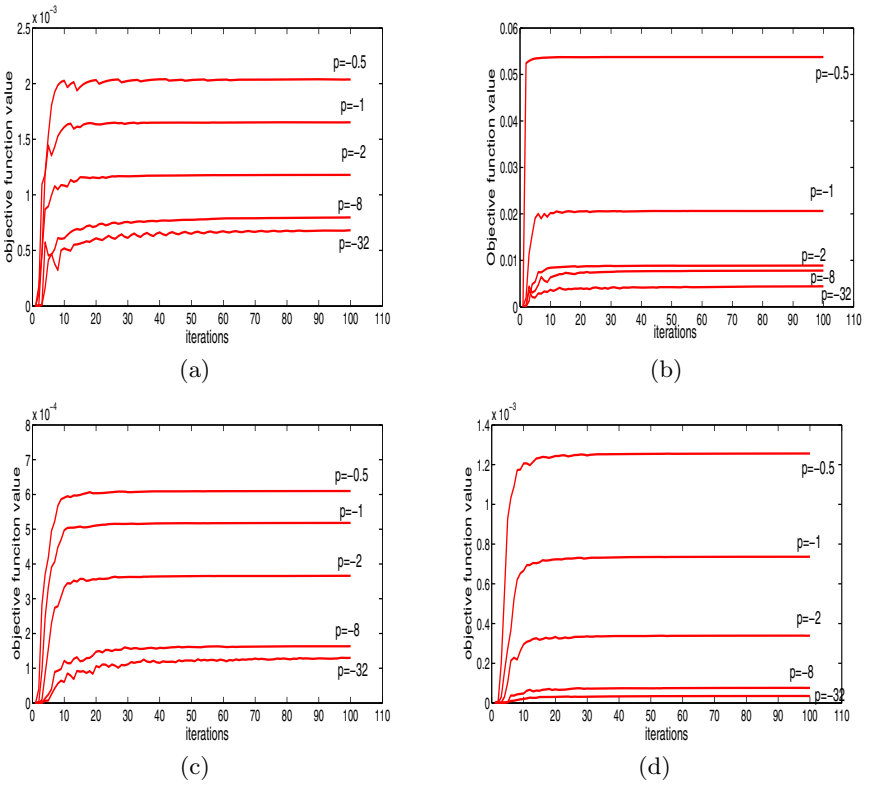


Fig. 1. Evolution of the objective function value of DML_p versus the number of iteration with varying p on Balance (a), Iris (b), Diabetes (c) and Image (d)

face images of 5749 people, 1680 of them appear in more than two images. There are two separate settings for forming training data: image-restricted and image-unrestricted setting. In the image-restricted paradigm, only the information whether a pair of images belongs to the same person (same class) is available and no information of actual names (class labels) in the pair of images is given. In the unrestricted setting, all available data including the identity of the people in the image is known. In this paper, we mainly consider the image-restricted setting.

The images we used are in gray scale and aligned in two ways. One is "funneled" [10] and the other is "aligned" using a commercial face alignment software [14]. We investigated several facial descriptors (features extracted from face images): 1) raw pixel data by concatenating the intensity value of each pixel in the image denoted by *Intensity*; 2) Local Binary Patterns (LBP) [13]; 3) Three-Patch Local Binary Patterns (TPLBP) [24]. For a fair comparison with [7], we

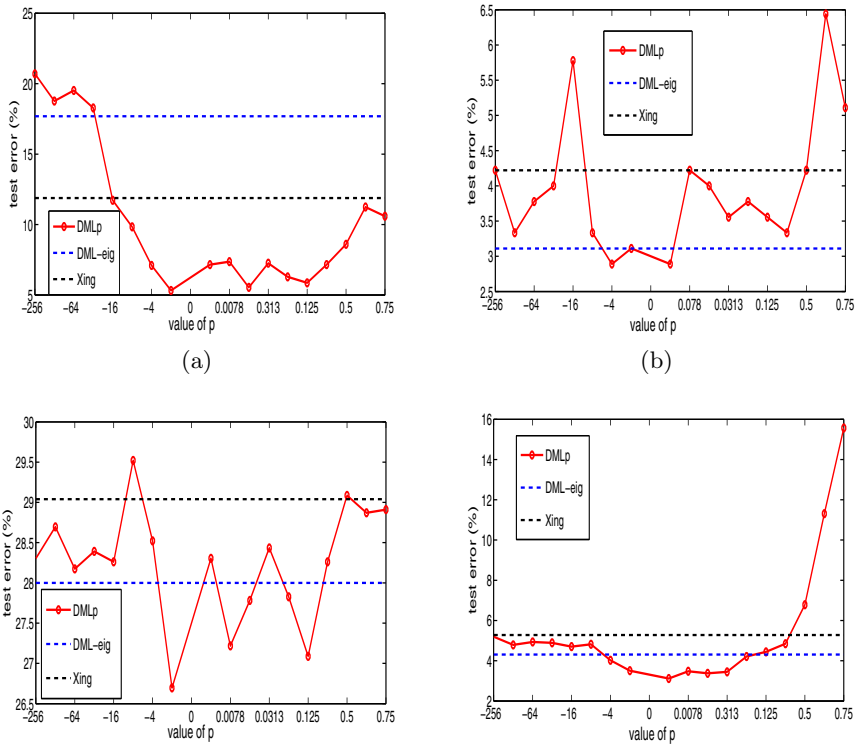


Fig. 2. Test error (%) of DML$_p$ versus different values of p on Balance (a), Iris (b), Diabetes (c) and Image (d). Red circled line is the result of DML$_p$ across different values of p (log-scaled); blue dashed line is the result of DML-eig and black dashed line represents the result of Xing.

also used SIFT descriptors [1] computed at the fixed facial key-points (e.g., corners of eyes and nose). Since the original dimensionality of the features is quite high (from 3456 to 12000), we reduced the dimension using principal component analysis (PCA). These descriptors were tested with both their original values and the square root of them [24,7].

In the image-restricted protocol, only pairwise constraints are given. LMNN and BoostMetric are not applicable to this setting since they require relative constraints in the form of triplets. Hence, we only compared our DML$_p$ method with ITML [4] and LDML [7]. The performance of our method is measured by the 10-fold cross-validation test. In each repeat, nine folds containing 2700 similar pairs of images and 2700 dissimilar pairs of images are used to learn

[1] http://lear.inrialpes.fr/people/guillaumin/data.php

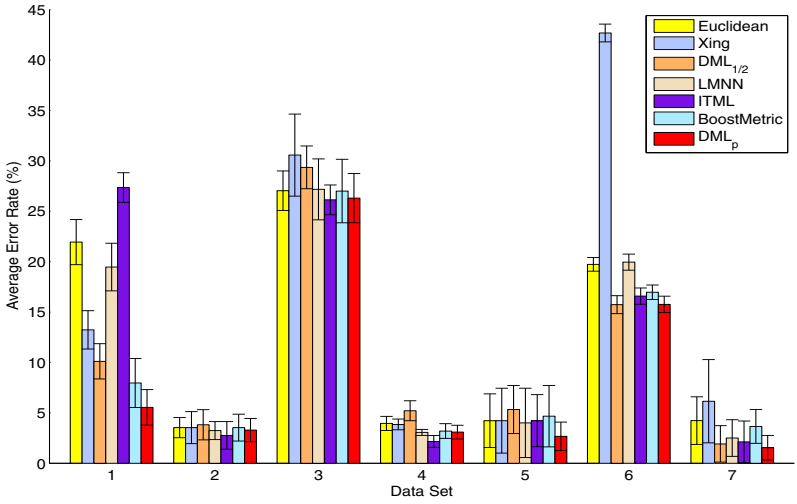


Fig. 3. Average test error (%) of DML_p against other methods

a metric and the remaining fold containing 600 image pairs is used to evaluate the performance of the metric learning method using accurate verification rate.

Firstly, we investigate the performance of DML_p on the SIFT descriptor by varying the dimension of principal components. Figure 4 depicts the verification accuracy versus the dimension of PCA. We can see that, compared to the ITML and LDML algorithms, our DML_p method using only SIFT descriptor delivers relatively stable performance as the PCA dimension varies. In particular, the performance of DML_p becomes stable after the dimension of PCA reaches around 100 and it consistently outperforms ITML across different PCA dimensions. We also observed similar results for other descriptors. Hence, for simplicity we set the PCA dimension to be 100 for the SIFT descriptor and other descriptors. According to [7], the best performances of LDML and ITML on the SIFT descriptor are 77.50% and 76.20% respectively. The best performance of DML_p reaches around 80% which outperforms ITML and LDML. We also note that the performance of ITML we got here is consistent with that reported in [7].

Secondly, we test the performance of our method using different descriptors and their combinations. Table 3 summarizes the results. In Table 3, the notation "Above combined" means that we combine the distance scores from the above listed (six) descriptors in the table using a linear Support Vector Machine (SVM), following the procedure in [7]. "All combined" means that all eight distance scores are combined. We observe that combining 4 descriptors (Intensity, SIFT, LBP and TFLBP) and their square-root ones yields 86.07% which outperforms

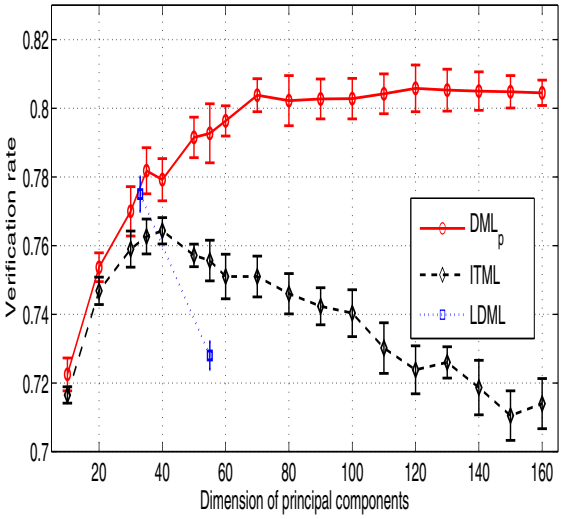


Fig. 4. Average verification rate of DML_p , ITML, and LDML on LFW by varying PCA dimension using the SIFT descriptor. The result of LDML is copied from Guillaumin et al. [7]: the best performance of LDML and ITML on the SIFT descriptor are respectively 77.50% and 76.20%.

Table 3. Performance of DML_p on LFW database with different descriptors (average verification accuracy and standard error). "DML_p SQRT" means DML_p uses the square root of the descriptor. "Intensity" means the raw pixel data by concatenating the intensity value of each pixel in the image. For all feature descriptors, the dimension is reduced to 100 using PCA. See more details in the text.

	DML_p	DML_p SQRT
SIFT	0.8015 ± 0.0055	0.8028 ± 0.0059
LBP	0.7972 ± 0.0062	0.8005 ± 0.0081
TPLBP	0.7790 ± 0.0058	0.7822 ± 0.0061
Above combined	0.8572 ± 0.0055	
Intensity	0.7335 ± 0.0054	0.7348 ± 0.0051
All combined	$\mathbf{0.8607 \pm 0.0058}$	

85.65% of DML-eig [28]. As mentioned above, DML-eig can be regarded as a limiting case of DML_p as $p \to -\infty$. This observation also validates the value of the general formulation DML_p . From Table 3, we can see that, although the individual performance of Intensity is inferior to those of other descriptors, combining it with other descriptors slightly increases the overall performance from 85.72% to 86.07%.

Finally, we summarize the performance of DML_p and other state-of-the-art methods in Table 4 and plot the ROC curve of our method compared to other

Table 4. Comparison of DML_p with other state-of-the-art methods in the restricted configuration (mean verification rate and standard error of the mean of 10-fold cross validation test) based on combination of different types of descriptors

Method	Accuracy
High-Throughput Brain-Inspired Features, aligned [18]	$\mathbf{0.8813 \pm 0.0058}$
LDML + Combined, funneled [7]	0.7927 ± 0.0060
DML-eig + Combined [28]	0.8565 ± 0.0056
DML_p + Combined (this work)	0.8607 ± 0.0058

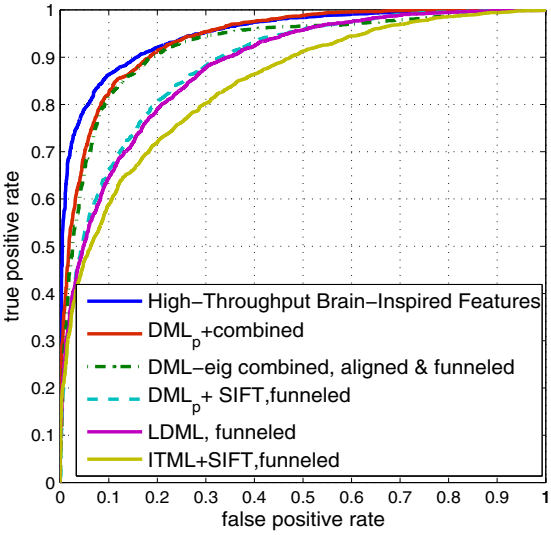


Fig. 5. ROC curves of DML_p and other state-of-the-art methods on LFW dataset

published results in Figure 5. We observe from Table 4 that our method DML_p outperforms LDML [7] and slightly improves the result of DML-eig [28]. The best performance on the restricted setting to date is 88.13% [18]. Note that the results compared here are system to system where metric learning is only one part of the system. We should also point out the result in [18] was not achieved by metric learning method. Instead, it performs sophisticated large scale feature search which used multiple complimentary representations derived through training set augmentation, alternative face comparison functions, and feature set searches with a varying number of model layers. We believe that the performance of DML_p may be further improved by exploring different types of descriptors such as those used in [18].

6 Conclusion

In this paper we extended and developed the metric learning models proposed in [25,28]. In particular, we proposed a general and unified framework which recovers the models in [25,28] as special cases. This novel framework was shown to be equivalent to a semi-definite program over the spectrahedron. This equivalence is important since it enables us to directly apply the Frank-Wolfe algorithm (e.g. [5,8]) to obtain the optimal solution. Experiments on UCI datasets validate the effectiveness of our proposed method and algorithm. In addition, the proposed method performs well on the Labeled Faces in the Wild (LFW) dataset in the task of face verification.

We now discuss some possible future work. It would be interesting to investigate the kernelised version of DML_p using similar ideas from [11,15]. Metric learning can be also regarded as a dimension reduction method. However, in its application to face verification, a common approach is to use PCA to reduce the dimensionality of the original descriptor. This triggers a natural question for future work on how to design effective metric learning methods to directly deal with the original descriptors of the images.

Acknowledgements. This work is supported by the EPSRC under grant EP/J001384/1. The corresponding author is Yiming Ying.

References

1. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a mahalanobis metric from equivalence constraints. Journal of Machine Learning Research 6, 937–965 (2005)
2. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively with application to face verification. In: CVPR (2005)
3. Cox, T., Cox, M.: Multidimensional scaling. Chapman and Hall, London (1994)
4. Davis, J., Kulis, B., Jain, P., Sra, S., Dhillon, I.: Information-theoretic metric learning. In: ICML (2007)
5. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Research Logistics Quaterly 3, 149–154 (1956)
6. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood component analysis. In: NIPS (2004)
7. Guillaumin, M., Verbeek, J., Schmid, C.: Is that you? Metric learning approaches for face identification. In: ICCV (2009)
8. Hazan, E.: Sparse Approximate Solutions to Semidefinite Programs. In: Laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L. (eds.) LATIN 2008. LNCS, vol. 4957, pp. 306–316. Springer, Heidelberg (2008)
9. Horn, R.A., Johnson, C.R.: Topics in Matrix Analysis. Cambridge University Press (1991)
10. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled Faces in the Wild: A database for studying face recognition in unconstrained environments. University of Massachusetts, Amherst, Technical Report 07-49 (2007)

11. Jain, P., Kulis, B., Dhillon, I.S.: Inductive regularized learning of kernel functions. In: NIPS (2010)
12. Jin, R., Wang, S., Zhou, Y.: Regularized distance metric learning: theory and algorithm. In: NIPS (2009)
13. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7), 971–987 (2002)
14. Taigman, Y., Wolf, L., Hassner, T.: Multiple one-shots for utilizing class label information. In: The British Machine Vision Conference (2009)
15. Tsang, I.W., Kwok, J.T.: Distance Metric Learning with Kernels. In: Kaynak, O., Alpaydın, E., Oja, E., Xu, L. (eds.) ICANN 2003 and ICONIP 2003. LNCS, vol. 2714. Springer, Heidelberg (2003)
16. Tenenbaum, J., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323 (2000)
17. Roweis, S.T., Lawrance, K.S.: Nonlinear dimensionality reduction by locally linear embedding. Science 290, 2323–2326 (2000)
18. Pinto, N., Cox, D.: Beyond simple features: a large-scale feature search approach to unconstrained face recognition. In: International Conference on Automatic Face and Gesture Recognition (2011)
19. Shen, C., Kim, J., Wang, L., Hengel, A.: Positive semidefinite metric learning with boosting. In: NIPS (2009)
20. Torresani, L., Lee, K.: Large margin component analysis. In: NIPS (2007)
21. Vandenbergheand, L., Boyd, S.: Semidefinite programming. SIAM Review 38(1), 49–95 (1996)
22. Weinberger, K.Q., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbour classification. In: NIPS (2006)
23. Weinberger, K.Q., Saul, L.K.: Fast solvers and efficient implementations for distance metric learning. In: ICML (2008)
24. Wolf, L., Hassner, T., Taigman, Y.: Descriptor based methods in the wild. In: Workshop on Faces Real-Life Images at ECCV (2008)
25. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance metric learning with application to clustering with side information. In: NIPS (2002)
26. Yang, L., Jin, R.: Distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering, Michigan State University (2007)
27. Ying, Y., Huang, K., Campbell, C.: Sparse metric learning via smooth optimization. In: NIPS (2009)
28. Ying, Y., Li, P.: Distance metric learning with eigenvalue optimization. Journal of Machine Learning Research 13, 1–26 (2012)

Geodesic Analysis on the Gaussian RKHS Hypersphere

Nicolas Courty[1,3], Thomas Burger[2], and Pierre-François Marteau[1]

[1] IRISA, Université de Bretagne Sud, Vannes, France
[2] iRTSV (FR3425) / BGE (U1038), CNRS/CEA/UJF/INSERM, Grenoble, France
[3] Institute of Automation, Chinese Academy of Science, Beijing, China

Abstract. Using kernels to embed non linear data into high dimensional spaces where linear analysis is possible has become utterly classical. In the case of the Gaussian kernel however, data are distributed on a hypersphere in the corresponding Reproducing Kernel Hilbert Space (RKHS). Inspired by previous works in non-linear statistics, this article investigates the use of dedicated tools to take into account this particular geometry. Within this geometrical interpretation of the kernel theory, Riemannian distances are preferred over Euclidean distances. It is shown that this amounts to consider a new kernel and its corresponding RKHS. Experiments on real publicly available datasets show the possible benefits of the method on clustering tasks, notably through the definition of a new variant of kernel k-means on the hypersphere. Classification problems are also considered in a classwise setting. In both cases, the results show improvements over standard techniques.

1 Introduction

Most of the well known methods using the kernel trick [1,2] postulate that since the data are embedded in a Kernel Reproducing Hilbert Space (RKHS) with high dimensionality, non-linear data description is likely to become linear. As such, most of the classical linear methods can be applied with benefits. However, in the RKHS associated to numerous kernels (including the Gaussian kernel, on which this work is focused), all vectors have a unitary norm: the dataset lies on a hypersphere [3]. Hence, should this particular geometry be explicitly exploited by using non linear statistical tools in the RKHS? This work is a step in this direction. We notably show on two different applications (classification and clustering) that this idea can yield enhanced results over some real world datasets. The key idea is to consider a geodesic distance on the hypersphere rather than the Euclidean one to perform the data analysis. The geodesic distance corresponds to the total length of the shortest path over the hypersphere between two points, and it can be computed readily using trigonometric operators (Figure 1). Interestingly enough, this leads us to the definition of a new kernel: It appears that the geodesic distances in the original RKHS are equivalent to the Euclidean distances in a new RKHS. Thus, when data are embedded in this latter, it is indeed really justified to use linear methods. Our construction can

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 299–313, 2012.

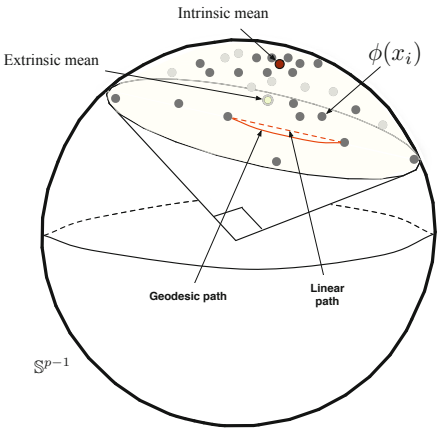


Fig. 1. Whatever the distribution of X, $\phi(X)$ lies within sphere quadrant. We propose to consider geodesic distance between elements of $\phi(X)$ rather than the Euclidean one. The Karcher (intrinsic) mean of $\phi(X)$ is represented as a red point, whereas the extrinsic mean is depicted in green. Note the latter is inside the hypersphere, whereas the Karcher mean lies on it.

be related to the work of Lafferty and Lebanon [4], who define a family of kernels based on diffusion operators over a Riemannian manifold. In our case, the geometric structure of the manifold is directly used to give a closed-form kernel expression instead of using a Fischer information metric.

The article is organized as follows: In Section 2, we set notations, and we provide background materials on geodesic distances and Riemannian manifolds. In Section 3 we adapt the classical tools of geodesic analysis to the Gaussian RKHS: To overcome the main drawback of kernelized space (the coordinates of the vectors are unknown), we find a transformation of the Gram matrix induced by the Gaussian kernel which takes into account geodesic distances. Next, in Section 4, we derive from the Gaussian kernel and from its modified Gram matrix a new data-dependent kernel. At this point, we remark that this derivation does not stand only for the Gaussian kernel, but for numerous other Radial Basis Function (RBF) kernels, leading to a whole family of data-dependent kernels. These latter are proved to be interesting on real datasets in Section 5: First, a clustering task is achieved by considering a k-means algorithm with geodesic distances on the Gaussian hypersphere. Second, we compare our new kernel to the Gaussian one in a classification task.

2 Geodesic Analysis on the Hypersphere

This section introduces the basis of a geodesic analysis on the hypersphere in the RKHS induced by the Gaussian Kernel. After stating the problem, basic facts about Riemannian geometry are presented and the notion of geodesic analysis is introduced.

2.1 Problem Statement

Let $X = \{x_1, \ldots, x_p\}_{(x_i \in \mathbb{R}^n)}$ be a set of p separated training samples described with n variables, and living in a space isomorphic to $\mathbb{R}^n$ and referred to as the *input space*. It is endowed with the Euclidean inner product denoted $\langle ., . \rangle_{\mathbb{R}^n}$ in the following. Let $k(.,.)$ be a symmetric form measuring the similarity among pairs of X, also called *kernel*. Let $\mathcal{H}$ be the associated RKHS, or *feature space*, also equipped with a dedicated inner product noted $\langle ., . \rangle_{\mathcal{H}}$, such that for any pair $(x_i, x_j) \in X^2$, we have:

$$\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}} = k(x_i, x_j) \tag{1}$$

where $\phi(.)$ is an implicit mapping from $\mathbb{R}^n$ onto $\mathcal{H}$. We use the shorthand notation $\phi(X)$ for the set $\{\phi(x_1), \ldots, \phi(x_p)\}_{(\phi(x_i) \in \mathcal{H})}$. $\mathbf{K}$ is the Gram matrix of $\phi(X)$, and as such $\mathbf{K}_{ij} = k(x_i, x_j)$. We use the generic notation x for any vector of $\mathbb{R}^n$. Similarly, any vector of $\mathcal{H}$ is noted $\phi(x)$ (if its pre-image is assumed to be x) or simply y (if there is no assumption on its pre-image).

A kernel of particular interest in this work is the Gaussian kernel, defined as:

$$k(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \tag{2}$$

with the variance parameter $\sigma^2 \in \mathbb{R}_+^*$. Remark that: (1) the norm of any $\phi(x_i) \in \mathcal{H}$ is the unity, *i.e.* $\langle \phi(x_i), \phi(x_i) \rangle_{\mathcal{H}} = 1$, (2) the Gaussian RKHS is of infinite dimension. As a consequence, whatever X, $\phi(X)$ spans a subspace of dimension exactly p, and as such $\phi(X)$ lies on the unit hypersphere $\mathbb{S}^{p-1} \subset \mathcal{H}$. Moreover, as the inner product of two unit vectors corresponds to the cosine of their angle, and as $\forall (x_i, x_j),\ k(x_i, x_j) \in [0, 1]$, whatever X, $\phi(X)$ lies in a restriction $\mathcal{R}$ of $\mathbb{S}^{p-1}$ which is embedded in a sphere quadrant (its maximum angle is smaller than or equal to $\pi/2$, such as illustrated on Figure 1). Naturally, as $k(x_i, x_j)$ varies according to the value of the σ parameter, the surface of $\mathcal{R}$ varies accordingly: When σ increases, $k(x_i, x_j)$ increases, (*i.e.* the cosine between x_i and x_j increases), and thus the surface of $\mathcal{R}$ decreases. Conversely, when $\sigma \to 0$, $\mathcal{R}$ tends to a sphere quadrant.

2.2 Analysis on Riemannian Manifolds

A Riemannian manifold $\mathcal{M}$ in a vector space $\mathcal{V}$ with inner product $\langle ., . \rangle_{\mathcal{V}}$ is a real differentiable manifold such that the tangent space $\mathcal{T}_{x^*}$ associated to each vector x^* is endowed with an inner product $\langle ., . \rangle_{\mathcal{T}_{x^*}}$. In this work, $\langle ., . \rangle_{\mathcal{T}_{x^*}}$ reduces to $\langle ., . \rangle_{\mathcal{V}}$ on $\mathcal{T}_{x^*}$, so for simplicity we assimilate $\langle ., . \rangle_{\mathcal{T}_{x^*}}$ to $\langle ., . \rangle_{\mathcal{V}}$.

Classically data analysis is performed in $\mathcal{V} = \mathbb{R}^n$ and not in $\mathcal{M}$, as in the former it is rather natural to formalize the intuitive geometric notions (distance, mean, variance, direction, etc.) which are necessary to characterize the dataset. On the other hand, the statistical analysis of a dataset within $\mathcal{M}$ requires the non-trivial generalization of these notions to the setting of Riemmanian geometry. One of the first statistical analysis tool designed for Riemannian manifold

is the Principal Geodesic Analysis (or PGA), the goal of which is to find a set of directions, called *geodesic directions* or *principal geodesics*, that best encode the statistical variability of the data. PGA was first introduced by Fletcher et al. [5], and received since then numerous addenda [6,7], which are beyond the scope of this work. Here, we only focus on the tools of Riemannian geometry which are involved in the definition of PGA. The crucial observation of Fletcher is that a first order approximation of the distances among the samples of the dataset can be obtained if one projects the dataset in $\mathcal{T}_\mu$, the tangent space at μ, the Karcher mean of the dataset. We recall that the *Karcher mean* [8] $\mu \in \mathcal{M}$ differs from the traditional mean $\overline{x} \in \mathcal{V}$ (also called the *extrinsic mean*): It is the point of $\mathcal{M}$ which minimizes the sum of squared geodesic distances to every input data. As such, it constitutes an *intrinsic mean* (see Figure 1 for an illustration). We have:

$$\mu = \arg\min_{x \in \mathcal{M}} \sum_{i=1}^{p} d_{geod}(x_i, x)^2. \tag{3}$$

This approximation of the geodesic distances in $\mathcal{M}$ by the Euclidean distances in $\mathcal{T}_\mu$ seems particularly appealing, and it has been shown [9] that for a sphere the induced error is rather low. However, as this manifold lies in $\mathcal{V} = \mathcal{H}$ (instead of $\mathbb{R}^n$), the tractability of this approximation addresses several questions: First, how to define geodesic distances on the manifold embedding $\phi(X)$, and compute the associated Karcher mean μ of $\phi(X)$? Second, how to characterize $\mathcal{T}_\mu$ and project $\phi(X)$ onto $\mathcal{T}_\mu$? These two questions are addressed in two dedicated subsections of the next section.

3 Data Analysis over the Hypersphere in the Gaussian RKHS

Let us consider the unit hypersphere $\mathbb{S}^{p-1} \in \mathcal{H}$, the surface of which is the Riemannian manifold which embeds $\phi(X)$.

3.1 Geodesic Distance and Karcher Mean

The Riemannian distance (or the geodesic distance) between $\phi(x_i)$ and $\phi(x_j)$ on $\mathbb{S}^{p-1}$ corresponds to the length of the portion of the great circle embedding $\phi(x_i)$ and $\phi(x_j)$. It is simply given by:

$$d_{geod}(\phi(x_i), \phi(x_j)) = \arccos(\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}). \tag{4}$$

Then Equation (3) reads:

$$\mu = \arg\min_{y \in \mathcal{H}} \sum_{i=1}^{p} \arccos(\langle \phi(x_i), y \rangle_{\mathcal{H}})^2. \tag{5}$$

The Karcher mean of X exists and is uniquely defined as long as X belongs to a Riemannian ball of radius $\pi/4$ [8,10] which is the case since two points can be

at maximum distant from $\pi/2$. Usually, non-linear optimization methods can be used to compute this mean. However, finding the coordinates for μ is impossible, since we do not have access to the coordinates of $\phi(X)$. Instead, we turn on the search of the pre-image $\tilde{x} \in \mathbb{R}^n$ of $\mu \in \mathcal{H}$ (such that $\mu = \phi(\tilde{x})$). It is the solution of the following (non-linear) minimization problem:

$$\tilde{x} = \arg\min_{x \in \mathbb{R}^n} \sum_{i=1}^{p} \arccos(\langle \phi(x_i), \phi(x) \rangle_{\mathcal{H}})^2, \tag{6}$$

$$= \arg\min_{x \in \mathbb{R}^n} \sum_{i=1}^{p} \arccos(k(x_i, x))^2. \tag{7}$$

To operate this minimization, let us consider

$$f : \mathbb{R}^n \to \mathbb{R}$$
$$x \mapsto \sum_{i=1}^{p} \arccos(k(x_i, x))^2$$

and compute its gradient:

$$\nabla f(x) = \sum_{i=1}^{p} \frac{\partial}{\partial x} \arccos(k(x_i, x))^2, \tag{8}$$

$$= \frac{2}{\sigma^2} \sum_{i=1}^{p} \frac{\arccos(k(x_i, x)) k(x_i, x)}{\sqrt{1 - k(x_i,\ x)^2}} (x_i - x).$$

Setting this derivative to zero leads to a fixed point algorithm similar to the seminal work on pre-image computation proposed by Mika et al. [11]. This algorithm amounts to refining in several iterations a solution $\tilde{x}^t$ such that:

$$\tilde{x}^{t+1} = \frac{\sum_i \alpha_t(i) x_i}{\sum_i \alpha_t(i)}, \tag{9}$$

with

$$\alpha_t(i) = \frac{\arccos(k(x_i, x)) k(x_i, \tilde{x}^t)}{\sqrt{1 - k(x_i, \tilde{x}^t)^2}}$$

However, as stated in [11], this approach is prone to find local minima and its output is strongly dependent on the choice of the initial guess. Therefore, we propose a simple greedy algorithm (Alg. 1), which simply consists in repeating p times the previous optimization by setting the initial guess as the different inputs x_i (this latter is then omitted in the sum of equation 9). The estimation of the Karcher's mean pre-image is achieved using Algorithm 1 with an $\mathcal{O}(k.n^2)$ complexity, where k is the number of iteration and n the number of samples. In practice k is small, namely less than 10 for the tested datasets when an RBF kernel is used. However, a possible drawback of this approach is that it only provides an approximation for the Karcher mean, since the true

Algorithm 1. Pre-image of the Karcher mean on the sphere in the RKHS

$\epsilon \leftarrow$ small value, $\tilde{x} \leftarrow \text{mean}(X)$
for $i = 1$ **to** p **do**
  $x_i^{t=0} \leftarrow x_i$
  repeat
    update $\tilde{x}_i^{t+1}$ using equation 9 with $\tilde{x}_i^t$
  until $||\tilde{x}_i^{t+1} - \tilde{x}_i^t||^2 < \epsilon$
  if $f(\tilde{x}_i^{t+1}) < f(\tilde{x})$ **then**
    $\tilde{x} \leftarrow \tilde{x}_i^{t+1}$
  end if
end for
Output $\tilde{x}$

one may not have an exact pre-image in the input space. Thus, it may be interesting to consider other approaches to find the pre-image of the Karcher mean, e.g. distance based [12] or local isomorphism [13]. Nevertheless, their direct application is impossible since the Karcher mean is only defined through a minimization procedure without a closed-form solution. Fig. 2 illustrates the result of Alg. 1 to compute the pre-image of the Karcher mean on two toy datasets (points randomly sampled over a square and a spiral in 2 dimensions).

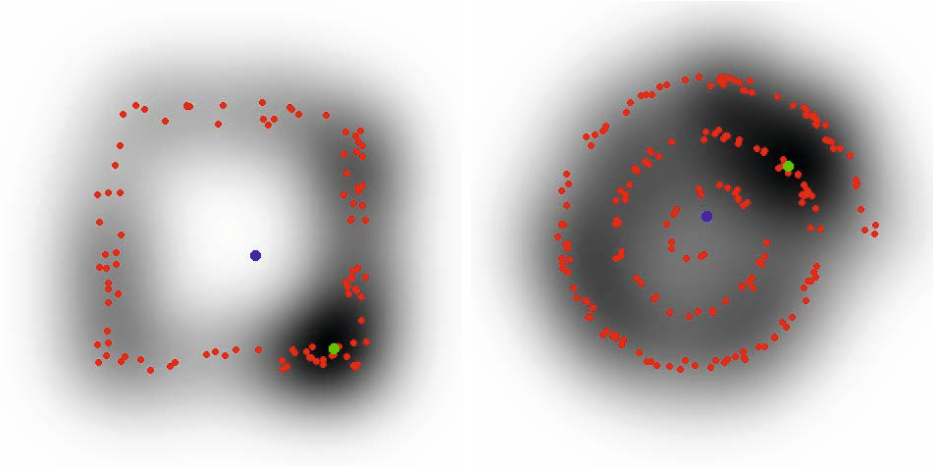


Fig. 2. Illustration of Karcher mean on two datasets: The dataset is represented by red points. The blue point is the data mean in input space, The green point is the pre-image of the Karcher mean after mapping onto the RKHS (the grayscale represents the function f values as described in Equation 8).

3.2 Projection on the Tangent Space

In the particular case of hypersperical manifolds, the mapping of any point onto a tangent space (this mapping is usually referred to as the *logarithmic map*), and the reverse mapping (the *exponential map*) are easy to define: The logarithmic map at location μ which projects any point $\phi(x_i) \in \mathcal{R} \subset \mathbb{S}^{p-1}$ onto $\mathcal{T}_\mu$ has the following form:

$$\begin{aligned} \text{Log}_\mu : \mathcal{R} \setminus \mu &\to \mathcal{T}_\mu \\ y &\mapsto \frac{\theta}{\sin(\theta)}(y - \cos(\theta) \cdot \mu) \end{aligned} \tag{10}$$

where θ is the angle between μ and y *i.e.* $\theta = \arccos(\langle \mu, y \rangle_{\mathcal{H}})$. When $\theta = 0$, it is natural to consider that $y = \mu$. Conversely, the exponential map[1], which projects a vector y of $\mathcal{T}_\mu$ onto $\mathbb{S}^{p-1}$, is defined as:

$$\begin{aligned} \text{Exp}_\mu : \mathcal{T}_\mu &\to \mathbb{S}^{p-1} \\ y &\mapsto \frac{\sin(\theta)}{\theta} \cdot y + \cos(\theta) \cdot \mu \end{aligned} \tag{11}$$

where θ is given by $\theta = \arccos\left(\frac{\langle y, \mu \rangle}{||y||}\right) = ||y||$.

When using the kernel notation, and for $\phi(x_i) \neq \mu$ Equation 10 reads:

$$\text{Log}_{\phi(\tilde{x})}(\phi(x_i)) = \frac{\arccos(k(x_i, \tilde{x}))}{\sqrt{1 - k(x_i, \tilde{x})^2}}(\phi(x_i) - k(x_i, \tilde{x})\phi(\tilde{x})). \tag{12}$$

So far, the exact computation of this projection cannot be conducted, as ϕ remains unknown. However, it is possible to derive the Gram matrix of $\text{Log}_{\phi(\tilde{x})}(\phi(X))$:

$$\begin{aligned} \mathbf{K}^{\tilde{x}}_{ij} &= \langle \text{Log}_{\phi(\tilde{x})}(\phi(x_i)), \text{Log}_{\phi(\tilde{x})}(\phi(x_j)) \rangle_{\mathcal{H}}, \\ &= \frac{\arccos(k(x_i, \tilde{x}))\arccos(k(x_j, \tilde{x}))}{\sqrt{1 - k(x_i, \tilde{x})^2}\sqrt{1 - k(x_j, \tilde{x})^2}} \cdot \\ &\quad (\phi(x_i) - k(x_i, \tilde{x})\phi(\tilde{x}))^T(\phi(x_j) - k(x_j, \tilde{x})\phi(\tilde{x})). \end{aligned} \tag{13}$$

Noting that:

$$\begin{aligned} &(\phi(x_i) - k(x_i, \tilde{x})\phi(\tilde{x}))^T(\phi(x_j) - k(x_j, \tilde{x})\phi(\tilde{x})) \\ &= \phi(x_i)^T\phi(x_j) - \phi(\tilde{x})^T\phi(x_j)k(x_i, \tilde{x}) - \phi(x_i)^T\phi(\tilde{x})k(x_j, \tilde{x}) + k(x_i, \tilde{x})k(x_j, \tilde{x}) \\ &= k(x_i, x_j) - 2k(x_i, \tilde{x})k(x_j, \tilde{x}) + k(x_i, \tilde{x})k(x_j, \tilde{x}) \\ &= k(x_i, x_j) - k(x_i, \tilde{x})k(x_j, \tilde{x}), \end{aligned} \tag{14}$$

we finally have a simple form for the entries of $\mathbf{K}^{\tilde{x}}$:

$$\mathbf{K}^{\tilde{x}}_{ij} = \frac{\arccos(k(x_i, \tilde{x}))\arccos(k(x_j, \tilde{x}))}{\sqrt{1 - k(x_i, \tilde{x})^2}\sqrt{1 - k(x_j, \tilde{x})^2}} \cdot (k(x_i, x_j) - k(x_i, \tilde{x})k(x_j, \tilde{x})). \tag{15}$$

Finally, it is possible to consider the geodesic distances in $\mathcal{H}$, by simply replacing the Gram matrix $\mathbf{K}$ associated to the kernel $k(.,.)$ by another Gram matrix $\mathbf{K}^{\tilde{x}}$.

[1] It is important to note that points on $\mathcal{R}$ are presented as vectors from the center of the hypersphere, while points on $\mathcal{T}_\mu$ are presented as vectors from μ.

4 A New Kernel Accounting for Geodesics in Hyperspherical RKHS

4.1 The Gaussian Case

However, it is possible to interpret $\mathbf{K}^{\tilde{x}}$ directly as the Gram matrix derived from a new kernel $k^{\tilde{x}}$ such that:

$$k^{\tilde{x}}(x_i, x_j) = \frac{\arccos(k(x_i, \tilde{x}))\arccos(k(x_j, \tilde{x}))}{\sqrt{1-k(x_i,\tilde{x})^2}\sqrt{1-k(x_j,\tilde{x})^2}} \cdot(k(x_i, x_j) - k(x_i, \tilde{x})k(x_j, \tilde{x})).$$

with the assumption that if $x_i = \tilde{x}$ (resp. x_j), then $k^{\tilde{x}}(x_i, x_j) = \arccos k(x_i, x_j)$. In such a perspective, we first need to establish the following result:

Proposition 1. *$k^{\tilde{x}}$ is a kernel.*

Proof:
First, let us prove that

$$\begin{aligned} k_1 : \mathbb{R}^n \times \mathbb{R}^n &\to \mathbb{R} \\ x_i, x_j &\mapsto k(x_i, x_j) - k(x_i, \tilde{x})k(x_j, \tilde{x}) \end{aligned}$$

is a kernel. To do so, let us simply consider

$$\begin{aligned} \Phi : \mathbb{R}^n &\to \mathcal{H} \\ x &\mapsto \phi(x) - k(x, \tilde{x})\phi(\tilde{x}) \end{aligned}$$

and remark that, obviously,

$$\begin{aligned} k_2 : \mathbb{R}^n \times \mathbb{R}^n &\to \mathbb{R} \\ x_i, x_j &\mapsto \Phi(x_i)^T \Phi(x_j) \end{aligned}$$

is a kernel, as k_2 corresponds to the Euclidean inner product in an another RKHS, onto which Φ maps. As it appears in Equation 14 that $k_1(x_i, x_j) = k_2(x_i, x_j)$, k_1 is also a kernel.

Second, remark that $k^{\tilde{x}}(x_i, x_j)$ can be re-written as the following conformal transformation $g(x_i)k_1(x_i, x_j)g(x_j)$ with:

$$\begin{aligned} g : \mathbb{R}^n &\to \mathbb{R} \\ x \neq \tilde{x} &\mapsto \frac{\arccos(k(x, \tilde{x}))}{\sqrt{1-k(x,\tilde{x})^2}} \\ \tilde{x} &\mapsto 1 \end{aligned}$$

which shows [3,14] that $k^{\tilde{x}}$ is a kernel and concludes the proof. □

Finally, we can consider a new vector space $\mathcal{H}^{\tilde{x}}$ with Euclidean inner product noted $\langle ., . \rangle_{\mathcal{H}^{\tilde{x}}}$ such that for any pair $x_i, x_j \in X^2$, we have:

$$\langle \phi^{\tilde{x}}(x_i), \phi^{\tilde{x}}(x_j) \rangle_{\mathcal{H}^{\tilde{x}}} = k^{\tilde{x}}(x_i, x_j)$$

with $\phi^{\tilde{x}}$ being the mapping from $\mathbb{R}^n$ onto $\mathcal{H}^{\tilde{x}}$. Then, the non-Euclidean distance in $\mathbb{R}^n$ derived from $k^{\tilde{x}}$ can be interpreted in two different ways: First, as the geodesic distances among $\phi(X)$ on the Gaussian RKHS, and according to a particular reference point $\tilde{x}$; Second, as the Euclidean distances among $\phi^{\tilde{x}}(X)$ on a new parametric RKHS $\mathcal{H}^{\tilde{x}}$ (with a data-dependent parameter $\tilde{x}$ corresponding to the pre-image of the Karcher mean of $\phi(X)$).

4.2 The General Case

Now, let us remark that these results stand not only for the Gaussian RKHS, on which our work is based, but also for any kernel which maps the dataset onto a hypersphere, and such as the angle between any pair of vectors is smaller than or equal to $\pi/2$.

This is notably the case for any "normalized" RBF kernel. Let us recall that a RBF kernel is of the form $k(x_i, x_j) = h(d(x_i, x_j))$ where d is a metric on $\mathbb{R}^n$ and where h is a function from $\mathbb{R}$ onto $\mathbb{R}^+$. By "normalized", we mean that

- $h(0) = 1$ (so that the vectors are of unit length in the RKHS, leading to a hyperspherical manifold)
- $h(x) \in [0, 1]\ \forall x \in \mathbb{R}$ (so that the dataset remains in a sphere quadrant)

5 Experiments

In this section, we assess the interest of geodesic analysis on hypersperical manifolds thanks to several experiments. First, we evaluate during a clustering task, the well-grounded of the use of geodesic distances and of the pre-image of Karcher mean. Then, during a second task involving supervised classification, we evaluate their interest through the kernel trick, as we compare our new kernel $k^{\tilde{x}}$, presented in Section 4.1, to the classical Gaussian kernel k. In both tests, we use a series of UCI datasets [15].

5.1 Hyperspherical Kernel Clustering

In this experiment, we propose to modify the kernel k-means procedure: First, the centroids of the clusters are computed as the pre-image of Karcher mean of each class, instead of the extrinsic mean. Second, the Euclidean distances are replaced by geodesic distances on the hypersphere of the Gaussian RKHS. Let us note that the distances are always computed between a centroid m_i and a sample x_j (*i.e.* between a pre-image in $\mathbb{R}^n$ and a vector in $\mathbb{R}^n$). Hence, even if this algorithm (see Algorithm 2) corresponds to a k-means in $\mathcal{H}^{\tilde{x}}$, there is no need to use the kernel trick with our new kernel: The geodesic distance simply reads $d_{geod}(x_j, m_i) = \arccos(k(x_j, m_i))$.

We compare this algorithm, that we call **hyperspherical clustering** to the classical k-means algorithm, its kernelized version [16] and to the spectral clustering algorithm described in [17]. From a qualitative point of view, let us remark

Algorithm 2. Hyperspherical clustering

Input: dataset X, number of clusters k
Output: k clusters
for $i = 1$ **to** k **do**
  Randomly initialize the m_i centroid of cluster i
end for
repeat
  for $j = 1$ **to** p **do**
    for $i = 1$ **to** k **do**
      Compute $d_{geod}(x_j, m_i) = \arccos(k(x_j, m_i))$
    end for
    $c_j = \arg\min_i d_{geod}(x_j, m_i)$
  end for
  Assign to m_i the Karcher mean of the set
    $\{x_\ell \in X / \ c_\ell = i\}$
until no more changes in the partitioning

that while the k-means does not requires the tuning of any parameter, the three other algorithms require the setting of σ to the proper value. Moreover, the traditional k-means and our version in $\mathcal{H}^{\tilde{x}}$ exhibit comparable complexities. They both are very light from a computational point of view with respect to the spectral clustering algorithm, as this latter requires the computation of the spectrum of a $p \times p$ matrix, which is $\mathcal{O}(p^3)$-complex.

The results are presented in Figure 3 for various values of σ, and in Table 1 where the best accuracy rates over the σ's are given. For all evaluations, the experiments are repeated 20 times to limit the effect of the random initialization, and only the mean accuracy and variance over the repetitions is displayed. Apart from Ionosphere and Glass, on which we are slightly less efficient than respectively the spectral clustering and the kernel k-means algorithms, our algorithm appears to be the most accurate in peak performance. As suggested by Figure 3, the performances of Hyperspectral clustering is also rather stable over a range of sigma values. Of course these preliminary results call for a broader comparison with more data and other clustering approaches.

5.2 Classification

The main restriction of the new kernel $k^{\tilde{x}}$ is that it relies on a data-dependent parameter, $\tilde{x}$. As the pre-image of the Karcher mean, $\tilde{x}$ can be understood as a representative of the dataset X. Thus, if X is separated into several classes, there is little chances that $\tilde{x}$ fits as a good representative of all the classes (it may fall between several classes). Hence, we think $k^{\tilde{x}}$ is more adapted to generative (class-wise) algorithms, in which a dedicated Karcher mean is computed for each class.

As a consequence, we do not use $k^{\tilde{x}}$ with the state-of-the-art SVM [1], as it is a discriminative algorithm for which the computation of the Karcher mean is likely to be unadapted. Instead, we consider the PerTurbo algorithm [18], which

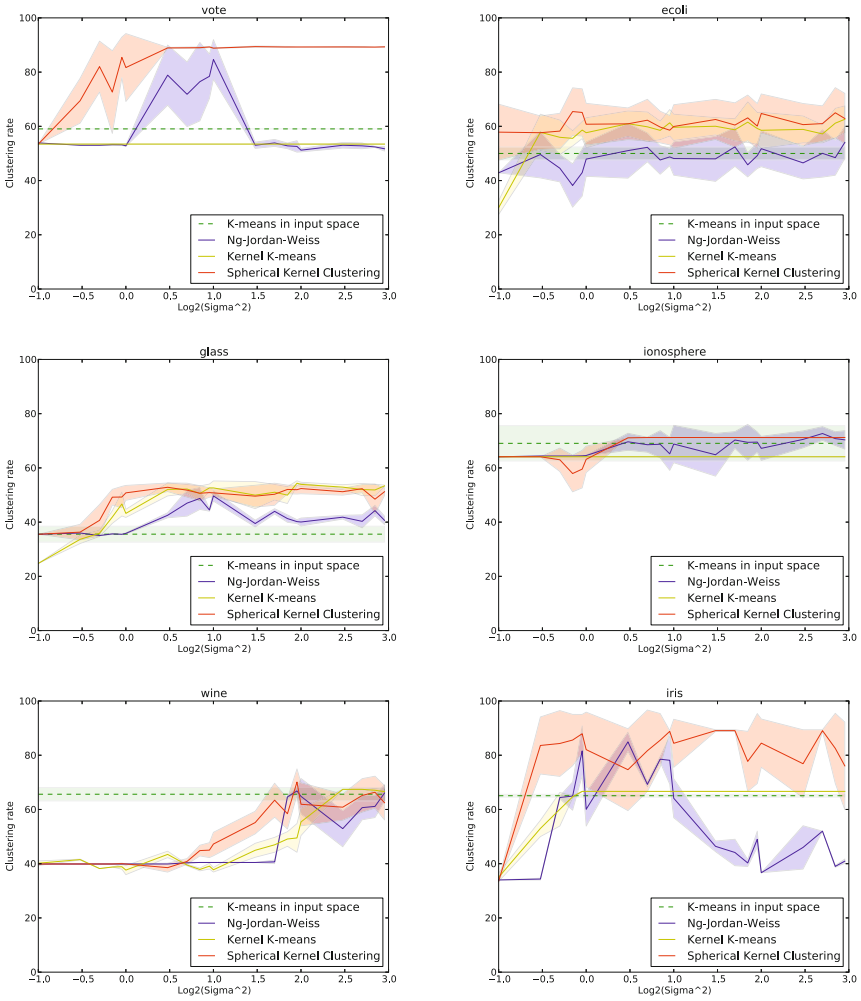


Fig. 3. Accuracy rates on the clustering task for different values of σ (on a logarithmic scale), and for the following algorithms: k-means, spectral clustering, kernel k-means and hypserspherical clustering

appears to provide similar performances while being generative. Nevertheless, let us note that our kernel would be adapted to one-class SVM for multi-class classification problems, such as in [19].

PerTurbo is a classification algorithm inspired from recent advances in computer graphics: Each class is characterized as a manifold in the input space (in a manner similar to the cloud of points giving birth to the 3D surface of a virtual object) thanks to an approximation of the Laplace-Beltrami operator [20,21]. As this approximation happens to be the Gaussian kernel, its perturbation measure (when a test sample is added to the manifold) can be re-interpreted in the

Table 1. Performances for the best σ. The last column indicates if our method outperforms the others.

Datasets	k-means	kernel k-means	spectral clustering	Hyperspherical clustering	Best ?
Iris	65.1 (0.5)	66.7 (0.0)	84.9 (3.6)	**89.1**(**0.3**)	√
Ecoli	50.0 (2.4)	62.6 (4.9)	54.1 (5.9)	**65.5**(**8.7**)	√
Ionosphere	69.1 (6.5)	64.1 (1.4)	**72.7**(**2.5**)	71.2 (0.0)	•
glass	35.6 (2.9)	**54.3**(**0.9**)	49.7 (1.9)	52.9 (1.6)	•
vote	59.1 (0.0)	53.4 (0.0)	84.7 (7.3)	**89.4**(**0.21**)	√
wine	65.6 (2.5)	67.4 (0.0)	66.8 (1.4)	**70.2**(**4.8**)	√

kernel machine learning setting. Moreover, the perturbation measure appears to correspond to the reconstruction error in Kernel-PCA. Hence, in a nutshell, PerTurbo can be interpreted as the following algorithm: (1) For each class, learn a set of eigenfunctions thanks to Kernel-PCA; (2) project any test sample onto the subspace associated to each class; (3) classify the test sample into the class for which the distance between the projection into the corresponding subspace and the sample is the smallest. Thus, PerTurbo can be related to some particular cases of subspace classifiers [22] in kernelized space.

This interpretation of PerTurbo in the kernel machines setting allows to use other kernels than the Gaussian one, whereas this latter is the only one which is proved to approximate the Laplace-Beltrami operator. Hence, we compare the result of PerTurbo with (1) the Gaussian kernel, (2) our new kernel. In addition, in order to remain comparable with more classical results of the state of the art, we also compare the results with C-SVM (using the `R` package kernlab [23]) using Gaussian kernel only. For the three algorithms, the experimental conditions are identical: the training set is made of 50% of the dataset randomly picked up, the process is repeated 30 times and the mean accuracy and its standard deviation are considered. The optimal value for the σ parameter is found with a logarithmic grid-search. For SVM, we did not use a grid-search for the C parameter, in order to make sure that the three algorithms have the same number of degrees of freedom which are fixed through a grid-search. Hence, following [3], we automatically tune C such that it is 10 times the number of training samples involved. In a multiclass setting, we consider the average number of training samples per class, *i.e.*: C = (Total number of training samples $\times$ 5)/Number of labels. Although this rule is very efficient, it only provides nearly optimal results, which may explain why on some datasets, there are little differences with the state-of-the-art accuracy. On the other hand, if one wants to fully optimize the value of C so that the regularization of the separating hyperplane is completely controlled, it is also possible to introduce an additional regularization parameter for PerTurbo and to tune it similarly to C [18]. The results are given in Table 2. The performances of PerTurbo are slightly lower than the SVM. However, this is advantageously balanced on more than half of the datasets by the use of a kernel accounting for geodesic distances. As a consequence, it appears that the

Table 2. Description of performances for classification. Mean classification accuracy rates for Perturbo with Gaussian Kernel with Euclidean (second column) and with geodesic distances (third column). The last column indicates whether the geodesic distance leads to some improvements over the Euclidean distance. Results with SVM are also given for references. The value between parenthesis is the standard deviation.

Datasets	SVM	PerTurbo (Euclidean)	PerTurbo (geodesic)	Better
Ionosphere	94.0 (1.1)	92.5 (0.9)	94.2 (1.2)	$\surd$
Blood	76.2 (1.8)	76.9 (1.2)	77.5 (1.8)	$\surd$
Parkinsons	91.0 (2.5)	95.3 (3.3)	94.9 (1.0)	$\approx$
Iris	96.4 (1.6)	96.3 (1.4)	96.8 (1.0)	$\approx$
Haberman	74.6 (2.4)	75.2 (4.2)	73.7 (0.7)	$\bullet$
Glasses	66.0 (3.9)	62.1 (0.9)	68.8 (3.0)	$\surd$
Wines	97.2 (1.5)	84.7 (2.2)	96.8 (1.0)	$\surd$
Diabetes	76.9 (1.3)	75.0 (1.8)	73.1 (1.9)	$\bullet$
Australian	86.5 (1.2)	86.2 (1.0)	84.5 (1.6)	$\bullet$
German	75.5 (1.5)	70.5 (1.8)	72.0 (1.5)	$\surd$

use of such a kernel on generative classifiers may enhance the results up to the performances of SVM. On a more qualitative point of view, it is interesting to recall that when σ increases, the portion of the sphere embedding $\phi(X)$ reduces, so that both (1) the error due to the approximation of the geodesic distance on the tangent space, and (2) the difference between the geodesic distance and the Euclidean one, decrease. As a consequence, the difference of performances between the two kernels should vanish for very high values of σ, and the well-grounded of the use of geodesic distance (in spite of the approximation in the tangent space) appears when it induces better performances than the Gaussian Kernel for small values of σ. These two phenomena are displayed in Figure 4 (for the Blood dataset).

6 Conclusion and Discussion

This work is motivated by a new idea: Some kernels have the interesting property to map the data onto a portion of the unit hypersphere, and on such a Riemannian manifold, non-linear data description techniques may be more adapted than linear ones. Thus, we first show how to adapt tools from Riemannian geometry (geodesic distances, Karcher mean) to RKHS, and we establish on clustering experiments that this path of study is worthwhile, notably through a new adaptation of the k-means algorithm on the hypersphere. Moreover, we prove that considering first order approximation of geodesic distances in the tangent space of the manifold is equivalent to use another kernel derived from the original one: when using this new kernel which directly embeds the geometry of the hypersphere, it is natural to consider linear separability method. This is also assessed by experiments on classification tasks.

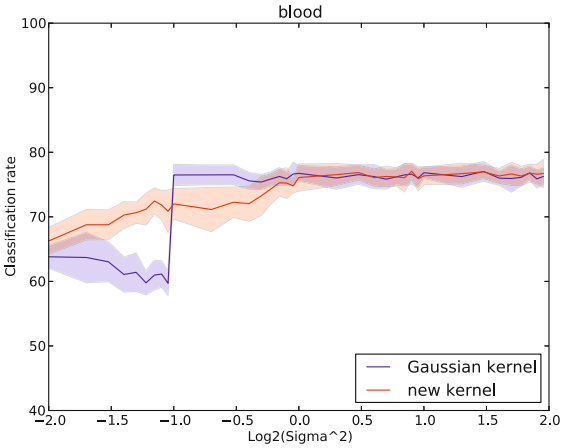


Fig. 4. Classification accuracy as a function of σ obtained on the Blood dataset. For large values of σ, accuracies of the two experimented kernels converge.

Although the proposed kernel that relies on the Karcher's mean is indeed data dependent, recent applications [19,24] seem to demonstrate that data dependent kernels may outperform data independent ones, provided that sufficient training data are available. This opens promising perspectives in multi-kernels learning, including the boosting of one-class SVM classifiers to address multi-class problems. This constitutes the most probable follow-up of this work.

Acknowledgments. The authors would like to thank the reviewers for their useful comments and remarks that helped in improving this paper. This work was supported by a Chinese Academy of Sciences visiting professorship for senior international scientists grant.

References

1. Cortes, C., Vapnik, V.: Support vector machine. Machine Learning 20(3), 273–297 (1995)
2. Schölkopf, B., Smola, A., Müller, K.R.: Kernel Principal Component Analysis. In: Gerstner, W., Hasler, M., Germond, A., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 583–588. Springer, Heidelberg (1997)
3. Schölkopf, B., Smola, A.J.: Learning with kernels: Support vector machines, regularization, optimization, and beyond. The MIT Press (2002)
4. Lafferty, J., Lebanon, G.: Diffusion kernels on statistical manifolds. Journal of Machine Learning Research 6, 129–163 (2005)
5. Fletcher, T., Lu, C., Pizer, S., Joshi, S.: Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Trans. Med. Imaging 23(8), 995–1005 (2004)
6. Said, S., Courty, N., LeBihan, N., Sangwine, S.J.: Exact principal geodesic analysis for data on so(3). In: Proceedings of EUSIPCO 2007, Poznan, Poland (2007)

7. Sommer, S., Lauze, F., Nielsen, M.: The differential of the exponential map, jacobi fields and exact principal geodesic analysis. CoRR, abs/1008.1902 (2010)
8. Karcher, H.: Riemannian center of mass and mollifier smoothing. Communications on Pure and Applied Mathematics 30(5), 509–541 (1977)
9. Sommer, S., Lauze, F., Hauberg, S., Nielsen, M.: Manifold Valued Statistics, Exact Principal Geodesic Analysis and the Effect of Linear Approximations. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 43–56. Springer, Heidelberg (2010)
10. Kendall, W.S.: Convexity and the hemisphere. Journal of the London Mathematical Society 2(3), 567 (1991)
11. Mika, S., Schölkopf, B., Smola, A.J., Müller, K.R., Scholz, M., Rätsch, G.: Kernel pca and de-noising in feature spaces. In: Advances in Neural Information Processing Systems, pp. 536–542. MIT Press (1999)
12. Kwok, J., Tsang, I.: The pre-image problem in kernel methods. IEEE Trans. on Neural Networks 15(6), 1517–1525 (2004)
13. Huang, D., Tian, Y., De la Torre, F.: Local isomorphism to solve the pre-image problem in kernel methods. In: CVPR 2011, pp. 2761–2768 (2011)
14. Amari, S.I., Wu, S.: Improving support vector machine classifiers by modifying kernel functions. Neural Networks 12(6), 783–789 (1999)
15. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
16. Dhillon, I., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: KDD, pp. 551–556 (2004)
17. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems 2, 849–856 (2002)
18. Courty, N., Burger, T., Laurent, J.: PerTurbo: A New Classification Algorithm Based on the Spectrum Perturbations of the Laplace-Beltrami Operator. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part I. LNCS, vol. 6911, pp. 359–374. Springer, Heidelberg (2011)
19. Chi-Yuan, Y., Zhi-Ying, L., Shie-Jue, L.: Boosting one-class support vector machines for multi-class classification. Applied Artificial Intelligence 23(4), 297–315 (2009)
20. Coifman, R.R., Lafon, S.: Diffusion maps. Applied and Computational Harmonic Analysis 21(1), 5–30 (2006)
21. Öztireli, C., Alexa, M., Gross, M.: Spectral sampling of manifolds. ACM Transaction on Graphics, Siggraph Asia (December 2010)
22. Cevikalp, H., Larlus, D., Neamtu, M., Triggs, B., Jurie, F.: Manifold based local classifiers: Linear and nonlinear approaches. Journal of Signal Processing Systems 61(1), 61–73 (2010)
23. Karatzoglou, A., Smola, A., Hornik, K., Zeileis, A.: kernlab-an s4 package for kernel methods in r (2004)
24. Gong, Y., Lazebnik, S.: Comparing data-dependent and data-independent embeddings for classification and ranking of internet images. In: CVPR, pp. 2633–2640. IEEE (2011)

Boosting Nearest Neighbors for the Efficient Estimation of Posteriors

Roberto D'Ambrosio[1,3], Richard Nock[2], Wafa Bel Haj Ali[3], Frank Nielsen[4], and Michel Barlaud[3,5]

[1] University Campus Bio-Medico of Rome, Rome, Italy
r.dambrosio@unicampus.it
[2] CEREGMIA - Université Antilles-Guyane, Martinique, France
rnock@martinique.univ-ag.fr
[3] CNRS - U. Nice, France
{belhajal,barlaud}@i3s.unice.fr
[4] Sony Computer Science Laboratories, Inc., Tokyo, Japan
Frank.Nielsen@acm.org
[5] Institut Universitaire de France

Abstract. It is an admitted fact that mainstream boosting algorithms like AdaBoost do not perform well to estimate class conditional probabilities. In this paper, we analyze, in the light of this problem, a recent algorithm, UNN, which leverages nearest neighbors while minimizing a convex loss. Our contribution is threefold. First, we show that there exists a subclass of surrogate losses, elsewhere called balanced, whose minimization brings simple and statistically efficient estimators for Bayes posteriors. Second, we show *explicit* convergence rates towards these estimators for UNN, for any such surrogate loss, under a Weak Learning Assumption which parallels that of classical boosting results. Third and last, we provide experiments and comparisons on synthetic and real datasets, including the challenging SUN computer vision database. Results clearly display that boosting nearest neighbors may provide highly accurate estimators, sometimes more than a hundred times more accurate than those of other contenders like support vector machines.

1 Introduction

Boosting refers to the iterative combination of classifiers which produces a classifier with reduced true risk (with high probability), while the base classifiers may be weakly accurate [1]. The final, *strong* classifier h, satisfies $\mathrm{im}(h) \subseteq \mathbb{R}$. Such an output carries out two levels of information. The simplest one is the sign of the output. This discrete value is sufficient to classify an unknown observation $\boldsymbol{x}$: $h(\boldsymbol{x})$ predicts that $\boldsymbol{x}$ belongs to a class of interest iff it is positive. The most popular boosting results typically rely on this sole information [1–3] (and many others). The second level is the real value itself, which carries out as additional information a magnitude which can be interpreted as a "confidence" in the classification. This continuous information may be fit into a link function

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 314–329, 2012.

$f : \mathbb{R} \to [0, 1]$ to estimate conditional class probabilities, thus lifting the scope of boosting to that of Bayes decision rule [4]:

$$\hat{\mathbf{Pr}}[y = 1|\boldsymbol{x}] = f(h(\boldsymbol{x})) \ . \tag{1}$$

To date, estimating posteriors with boosting has not met the same success as predicting (discrete) labels. It is widely believed that boosting and conditional class probability estimation are, up to a large extent, in conflict with each other, as boosting iteratively improves classification at the price of progressively overfitting posteriors [4, 5]. Experimentally, limiting overfitting is usually obtained by tuning the algorithms towards early stopping [6]. Very recently, a new algorithm was proposed to leverage the famed nearest neighbor (NN) rules [7, 8]. This algorithm, UNN, fits real-valued coefficients for examples in order to minimize a surrogate risk [2, 9]. These leveraging coefficients are used to balance the votes in the final k-NN rule. It is proven that, as the number of iterations $T \to \infty$, UNN achieves the global optimum of the surrogate risk at hand for a wide class of surrogates called strictly convex surrogates [2, 10]. An explicit convergence rate is obtained for the specific case of the exponential loss, under a so-called "weak index assumption" [8], generalized in [7]. Our contribution is threefold. First, we show that there exists a subclass of surrogate losses, elsewhere called *balanced*, whose minimization brings simple and efficient estimators for Bayes posteriors (11). Second, we show explicit convergence rates for UNN for *any* such surrogate loss under a Weak Learning Assumption which parallels that of classical boosting results [3]. Third and last, we provide experiments on simulated and real domains, displaying that boosting nearest neighbors brings very good results from the conditional class probabilities estimation standpoint, *without* the overfitting problem of classical boosting approaches. A serious challenger to the popular logistic estimator for posteriors estimation also emerges. We end up with the conclusion that learning posteriors with boosting nearest neighbors benefits from two advantages. First, the weak classifiers being simple examples, they naturally limit the risk of overfitting compared to more complex weak learners. Second, we end up learning posteriors using a *natural, fixed* topology of data, and not an *ad hoc* topology relying on an induced classifier.

The remaining of the paper is structured as follows: the next Section presents definitions, followed by a Section on convex losses and the estimation of posteriors. Then, a Section presents algorithms and results on boosting nearest neighbors. The two last Sections present experiments with discussions, and conclude.

2 Definitions

2.1 Estimation

Our setting is that of multiclass multilabel classification (See *e.g.* [3]). We have access to an input set of m examples, also called prototypes, $\mathcal{S} \doteq \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$. Vector $\boldsymbol{y}_i \in \{-1, 1\}^C$ encodes class memberships, assuming $y_{ic} = 1$

means that observation $\boldsymbol{x}_i$ belongs to class c. $\mathcal{S}$ is sampled i.i.d. according to an unknown distribution $\mathcal{D}$. Given an observation $\boldsymbol{x} \in \mathcal{O}$, we wish to estimate the conditional class probabilities for each class c, also called (estimated) posteriors:

$$\hat{p}_c(\boldsymbol{x}) \doteq \hat{\mathbf{Pr}}[y_c = 1|\boldsymbol{x}] \ . \tag{2}$$

We note $p_c(\boldsymbol{x}) \doteq \mathbf{Pr}_{\mathcal{D}}[y_c = 1|\boldsymbol{x}]$ the corresponding Bayes (true) posteriors.

2.2 Surrogates, Losses and Risks

Perhaps the simplest road towards computing these estimators consists in first crafting C separate classification problems, each of which leads to estimators for one class (2). Normalizing estimators to 1 over the C classes yields the values in (2). Each of these C problems is a one-versus-all classification task, say for class c, with corresponding sample $\mathcal{S}^{(c)} = \{(\boldsymbol{x}_i, y_{ic}), i = 1, 2, ..., m\}$. For each of these problems, we learn from $\mathcal{S}$ a classifier $h : \mathcal{O} \to \mathbb{R}$ out of which we may accurately compute (2), typically with $\hat{p}_c(\boldsymbol{x}) = f(h(\boldsymbol{x}))$ for some relevant function f. More sophisticated approaches exist that reduce the number of classifiers by folding classes in observation variables [3, 4]. Each of them equivalently learn on a sample of $\Omega(mC)$ examples, and it is an easy task to craft from their output a set of C classifiers that fit into the framework we consider.

There exists a convenient approach to carry out this path as a whole, for each class $c = 1, 2, ..., C$: learn h by minimizing a *surrogate risk* over $\mathcal{S}$ [2, 9, 10]. A surrogate risk has general expression:

$$\varepsilon^{\psi}_{\mathcal{S}}(h, c) \doteq \frac{1}{m} \sum_{i=1}^{m} \psi(y_{ic} h(\boldsymbol{x})) \ , \tag{3}$$

for some function ψ that we call a *surrogate loss*. Quantity $y_{ic} h(\boldsymbol{x}) \in \mathbb{R}$ is called the *edge* of classifier $\boldsymbol{h}$ on example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for class c. The surrogate risk is an estimator of the *true surrogate risk* computed over $\mathcal{D}$:

$$\varepsilon^{\psi}_{\mathcal{D}}(h, c) \doteq \mathbf{E}_{\mathcal{D}}[\psi(y_{ic} h(\boldsymbol{x}))] \ . \tag{4}$$

Any surrogate loss relevant to classification [9] has to meet $\text{sign}(h_{\text{opt}}(\boldsymbol{x}^*)) = \text{sign}(2\mathbf{Pr}_{\mathcal{D}}[y_c = 1|\boldsymbol{x} = \boldsymbol{x}^*] - 1)$, where h_{opt} minimizes $\mathbf{E}_{\mathcal{D}}[\psi(y_c h(\boldsymbol{x}))|\boldsymbol{x} = \boldsymbol{x}^\star]$. Hence, the sign of the optimal classifier h_{opt} is as accurate to predict class membership as Bayes decision rule. This Fisher consistency requirement for ψ is called *classification calibration* [9]. We focus in this paper on the subclass of classification calibrated surrogates that are strictly convex and differentiable.

Definition 1. *[2] A* ***strictly convex loss*** *is a strictly convex function ψ differentiable on* $\text{int}(\text{dom}(\psi))$ *satisfying (i)* $\text{im}(\psi) \subseteq \mathbb{R}^+$*, (ii)* $\text{dom}(\psi)$ *symmetric around* 0*, (iii)* $\nabla_\psi(0) < 0$*.*

Definition 1 is extremely general: should we have removed conditions (i) and (ii), Theorem 6 in [9] brings that it would have encompassed the intersection between

strictly convex differentiable functions and classification calibrated functions. Conditions (i) and (ii) are mainly conveniences for classification: in particular, it is not hard to see that modulo scaling by a positive constant, the surrogate risk (3) is an upperbound of the empirical risk for any strictly convex loss. Minimizing the surrogate risk amounts thus to minimize the empirical risk up to some extent. We define the Legendre conjugate of any strictly convex loss ψ as $\psi^\star(x) \doteq x\nabla_\psi^{-1}(x) - \psi(\nabla_\psi^{-1}(x))$. There exists a particular subset of strictly convex losses of independent interest [2]. A function $\phi : [0,1] \to \mathbb{R}^+$ is called *permissible* iff it is differentiable on $(0,1)$, strictly concave and symmetric around $x = 1/2$ [2, 11]. We adopt the notation $\overline{\phi} = -\phi$ [2].

Definition 2. *[2] Given some permissible ϕ, we let ψ_ϕ denote the* ***balanced convex loss*** *with signature ϕ as:*

$$\psi_\phi(x) \doteq \frac{\overline{\phi}^\star(-x) - \phi(0)}{\phi\left(1/2\right) - \phi(0)} \ . \tag{5}$$

Balanced convex losses have an important rationale: up to differentiability constraints, they match the set of symmetric lower-bounded losses defining proper scoring rules [2], that is, basically, the set of losses that fit to classification problems without class-dependent misclassification costs. Table 1 provides examples of surrogate losses, most of which are strictly convex surrogates, some of which are balanced convex surrogates. We have derived Amari's α-loss from Amari's famed α divergences [12] (proof omitted). The linear Hinge loss is *not* a balanced convex loss, yet it figures the limit behavior of balanced convex losses [2]. Remark that all signatures ϕ are well-known in the domain of decision-tree induction : from the top-most to the bottom-most, one may recognize Gini criterion, the entropy (two expressions), Matsushita's criterion and the empirical risk [10, 11].

2.3 One Dimensional Exponential Families and Posteriors Estimation

A (regular) one dimensional *exponential family* [12] is a set of probability density functions whose elements admit the following canonical form:

$$p[x|\theta] \doteq \exp\left(x\theta - \psi(\theta)\right) p_0(x) \ , \tag{6}$$

where $p_0(x)$ normalizes the density, ψ is a strictly convex differentiable function that we call the *signature* of the family, and θ is the density's natural parameter. It was shown in [2] that the efficient minimization of any balanced convex surrogate risk — *i.e.* a surrogate risk with a balanced convex loss — amounts to a maximum likelihood estimation $\hat{\theta} = H(\boldsymbol{x})$ at some $\boldsymbol{x}$ for an exponential family whose signature depends solely on the permissible function ϕ. [2] suggest to use the corresponding *expected* parameter of the exponential family as the posterior:

$$\hat{\mathbf{Pr}}[y = 1|\boldsymbol{x}] = \hat{\mathbf{Pr}}_\phi[y = 1|\boldsymbol{x}; H] \doteq \nabla_{\overline{\phi}}^{-1}(H(\boldsymbol{x})) \in [0,1] \ . \tag{7}$$

$\nabla_{\overline{\phi}}^{-1}$ plays the role of the link function (11). The quality of such an estimator shall be addressed in the following Section.

Table 1. Examples of surrogates ψ (Throughout the paper, we let ln denote the base-e logarithm, and $\log_z(x) \doteq \ln(x)/\ln(z)$ denote the base-z logarithm). From top to bottom, the losses are known as: squared loss, (normalized) logistic loss, binary logistic loss, Matsushita loss [2, 10], linear Hinge loss, exponential loss, Amari's α-loss, for $\alpha \in (-1, 1)$ [2]. Strictly convex losses are A, B, C, D, F, G. Balanced convex losses are A, B, C, D (E corresponds to a limit behavior of balanced convex losses [2]). For each ψ, we give the corresponding estimators $\hat{p}_c(\boldsymbol{x})$ (Theorem 1 and Eqs (9, 11) below: replace x by $h_{\text{opt}}(\boldsymbol{x})$), and if they are balanced convex losses, the corresponding concave signature ϕ (See text for details).

	ψ	$\hat{p}_c(\boldsymbol{x})$	ϕ
A	$(1-x)^2$	$\frac{1}{2}(1+x)$	$x(1-x)$
B	$\log_2(1+\exp(-x))$	$[1+\exp(-x)]^{-1}$	$\begin{array}{c}-x\ln x\\ -(1-x)\ln(1-x)\end{array}$
C	$\log_2(1+2^{-x})$	$\left[1+2^{-x}\right]^{-1}$	$\begin{array}{c}-x\log_2 x\\ -(1-x)\log_2(1-x)\end{array}$
D	$-x+\sqrt{1+x^2}$	$\frac{1}{2}\left(1+\frac{x}{\sqrt{1+x^2}}\right)$	$\sqrt{x(1-x)}$
E	$\frac{1}{2}x(\text{sign}(x)-1)$	$\begin{cases}1 & \text{if } x>0\\ 0 & \text{if } x<0\end{cases}$	$2\min\{x, 1-x\}$
F	$\exp(-x)$	$[1+\exp(-2x)]^{-1}$	N/A
G	$\left(1+\frac{1-\alpha^2}{4}x\right)^{-\frac{1+\alpha}{1-\alpha}}$	$\left[1+\left(\frac{4-(1-\alpha^2)x}{4+(1-\alpha^2)x}\right)^{\frac{2}{1-\alpha}}\right]^{-1}$	N/A

3 Strictly Convex Losses and the Efficient Estimation of Posteriors

There is a rationale to use (7) as the posterior: the duality between natural and expectation parameters of exponential families, via Legendre duality [2, 9], and the fact that the domain of the expectation parameter of one dimensional exponential families whose signature is (minus) a permissible function is the interval $[0, 1]$ [2]. We improve below this rationale, with the proof that *Bayes posteriors* satisfy (7) for the classifier which is the population minimizer of (7).

Theorem 1. *Suppose ψ strictly convex differentiable. The true surrogate risk $\boldsymbol{E}_{\mathcal{D}}[\psi(y_{ic}h(\boldsymbol{x}))]$ is minimized at the unique $h_{\text{opt}}(\boldsymbol{x})$ satisfying:*

$$\frac{\nabla_\psi(-h_{\text{opt}}(\boldsymbol{x}))}{\nabla_\psi(h_{\text{opt}}(\boldsymbol{x}))} = \frac{p_c(\boldsymbol{x})}{1-p_c(\boldsymbol{x})} \ . \tag{8}$$

Furthermore, is ψ is a balanced convex loss, then the population minimizer h_{opt} of $\boldsymbol{E}_{\mathcal{D}}[\psi_\phi(y_{ic}h(\boldsymbol{x}))]$ satisfies:

$$p_c(\boldsymbol{x}) = \nabla_\phi^{-1}(h_{\text{opt}}(\boldsymbol{x})) \ , \tag{9}$$

for which

$$\boldsymbol{E}_{\mathcal{D}}[\psi_\phi(y_{ic}h_{\text{opt}}(\boldsymbol{x}))] = \frac{\phi(p_c(\boldsymbol{x})) - \phi(0)}{\phi(1/2) - \phi(0)} \ . \tag{10}$$

(Proof omitted) Table 1 provides examples of expressions for $p_c(\boldsymbol{x})$ as in (9). Eq. (8) in Theorem (1) brings that we may compute an estimator $\hat{p}_c(\boldsymbol{x})$ as:

$$\hat{p}_c(\boldsymbol{x}) = \frac{\nabla_\psi(-h(\boldsymbol{x}))}{\nabla_\psi(h(\boldsymbol{x})) + \nabla_\psi(-h(\boldsymbol{x}))} \ . \tag{11}$$

This simple expression is folklore, at least for the logistic and exponential losses [4, 6]. The essential contribution of Theorem 1 relies on bringing a strong rationale to the use of (7), as the estimators converge to Bayes posteriors in the infinite sample case. Let us give some finite sample properties for the estimation (7). We show that the sample-wise estimators of (9) are efficient estimators of (9); this is not a surprise, but comes from properties of exponential families [13]. What is perhaps more surprising is that the corresponding aggregation of classifiers is not a linear combination of all estimating classifiers, but a generalized $\nabla_{\overline{\phi}}^{-1}$-mean.

Theorem 2. *Suppose we sample* n *datasets* $\mathcal{S}_j^{(c)}, j = 1, 2, ..., n$. *Denote* $\hat{h}_{\mathrm{opt},j}$ *the population minimizer for* $\boldsymbol{E}_{\mathcal{S}_j^{(c)}}[\psi_\phi(y_{ic}h(\boldsymbol{x}))]$. *Then each* $\hat{p}_{c,j}(\boldsymbol{x}) \doteq \nabla_{\overline{\phi}}^{-1}(\hat{h}_{\mathrm{opt},j}(\boldsymbol{x}))$ *is the only efficient estimator for* $p_c(\boldsymbol{x})$. *The corresponding classifier* $\hat{h}_{\mathrm{opt}}$ *aggregating all* $\hat{h}_{\mathrm{opt},j}$, *is:* $\hat{h}_{\mathrm{opt}}(\boldsymbol{x}) \doteq \nabla_{\overline{\phi}}\left(\frac{1}{n_{\boldsymbol{x}}}\sum_{j:(\boldsymbol{x},.)\in\mathcal{S}_j^{(c)}} \nabla_{\overline{\phi}}^{-1}(\hat{h}_{\mathrm{opt},j}(\boldsymbol{x}))\right), \forall \boldsymbol{x} \in \cup_j \mathcal{S}_j$, *where* $1 \leq n_{\boldsymbol{x}} \leq n$ *is the number of subsets containing* $\boldsymbol{x}$.

(Proof omitted)

4 Leveraging and boosting Nearest Neighbors

The nearest neighbor rule belongs to the oldest, simplest and most widely studied classification algorithms [14, 15]. We denote by $\mathrm{NN}_k(\boldsymbol{x})$ the set of the k-nearest neighbors (with integer constant $k > 0$) of an example $(\boldsymbol{x}, \boldsymbol{y})$ in set $\mathcal{S}$ with respect to a non-negative real-valued "distance" function. This function is defined on domain $\mathcal{O}$ and measures how much two observations differ from each other. This dissimilarity function thus may not necessarily satisfy the triangle inequality of metrics. For the sake of readability, we let $j \sim_k \boldsymbol{x}$ denote the assertion that example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ belongs to $\mathrm{NN}_k(\boldsymbol{x})$. We shall abbreviate $j \sim_k \boldsymbol{x}_i$ by $j \sim_k i$. To classify an observation $\boldsymbol{x} \in \mathcal{O}$, the k-NN rule $\mathcal{H}$ over $\mathcal{S}$ computes the sum of class vectors of its nearest neighbors, that is: $\mathcal{H}(\boldsymbol{x}) = \sum_{j\sim_k \boldsymbol{x}} \mathbf{1} \circ \boldsymbol{y}_j$, where $\circ$ is the Hadamard product. $\mathcal{H}$ predicts that $\boldsymbol{x}$ belongs to each class whose corresponding coordinate in the final vector is positive. A *leveraged* k-NN rule is a generalization of this to:

$$\mathcal{H}(\boldsymbol{x}) = \sum_{j\sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j \ , \tag{14}$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^C$ is a leveraging vector for the classes in $\boldsymbol{y}_j$. Leveraging approaches to nearest neighbors are not new [16, 17], yet to the best of our knowledge no

Algorithm 1. Algorithm UNIVERSAL NEAREST NEIGHBORS, UNN$(\mathcal{S}, \psi, k)$

Input: $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m,\ \boldsymbol{x_i} \in \mathcal{O},\ \boldsymbol{y_i} \in \{-1, 1\}^C\}$, ψ strictly convex loss (Definition 1), $k \in \mathbb{N}_*$;

Let $\boldsymbol{\alpha}_j \leftarrow \mathbf{0}, \forall j = 1, 2, ..., m$;

for $c = 1, 2, ..., C$ **do**

  Let $\boldsymbol{w} \leftarrow -\nabla_\psi(0)\mathbf{1}$;

  for $t = 1, 2, ..., T$ **do**

    [I.0] Let $j \leftarrow$ WIC$(\mathcal{S}, \boldsymbol{w})$;

    [I.1] Let $\delta_j \in \mathbb{R}$ solution of:

$$\sum_{i: j \sim_k i} y_{ic} y_{jc} \nabla_\psi \left(\delta_j y_{ic} y_{jc} + \nabla_\psi^{-1}(-w_i) \right) = 0 \ ; \tag{12}$$

    [I.2] $\forall i : j \sim_k i$, let

$$w_i \leftarrow -\nabla_\psi \left(\delta_j y_{ic} y_{jc} + \nabla_\psi^{-1}(-w_i) \right) \ , \tag{13}$$

    [I.3] Let $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$;

Output: $\mathcal{H}(\boldsymbol{x}) \doteq \sum_{j \sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j$

convergence results or rates were known, at least until the algorithm UNN [7, 8]. Algorithm 1 gives a simplified version of the UNN algorithm of [7, 8] which learns a leveraged k-NN. Oracle WIC$(\mathcal{S}, \boldsymbol{w})$ is the analogous for NN of the classical weak learners for boosting: it takes learning sample $\mathcal{S}$ and weights $\boldsymbol{w}$ over $\mathcal{S}$, and returns the index of some example in $\mathcal{S}$ which is to be leveraged. [8] prove that for any strictly convex loss ψ, UNN converges to the global optimum of the surrogate risk at hand. However, they prove boosting-compliant convergence rates only for the exponential loss. For all other strictly convex losses, there is no insight on the rates with which UNN may converge towards the optimum of the surrogate risk at hand. We now provide such explicit convergence rates under the following *Weak Learning Assumption*:

WLA: There exist some $\vartheta > 0, \varrho > 0$ such that, given any $k \in \mathbb{N}_*$, $c = 1, 2, ..., C$ and any distribution $\boldsymbol{w}$ over $\mathcal{S}$, the weak index chooser oracle WIC returns an index j such that the following two statements hold:
(i) $\mathbf{Pr}_{\boldsymbol{w}}[j \sim_k i] \geq \varrho$;
(ii) $\mathbf{Pr}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \sim_k i] \leq 1/2 - \vartheta$ or $\mathbf{Pr}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \sim_k i] \geq 1/2 + \vartheta$.

Requirement (i) is a weak *coverage* requirement, which "encourages" WIC to choose indexes in dense regions of $\mathcal{S}$. Before studying the boosting abilities of UNN, we focus again on surrogate risks. So far, the surrogate risk (3) has been evaluated with respect to a single class. In a multiclass multilabel setting, we may compute the *total* surrogate risk over all classes as:

$$\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H}) \doteq \frac{1}{C} \sum_{c=1}^{C} \varepsilon_{\mathcal{S}}^{\psi}(h_c, c) \ , \tag{15}$$

where $\mathcal{H}$ is the set of all C classifiers $h_1, h_2, ..., h_C$ that have been trained to minimize each $\varepsilon_{\mathcal{S}}^{\psi}(., c), c = 1, 2, ..., C$. We split classifiers just for convenience

in the analysis: if one trains a single classifier $H : \mathcal{O} \times \{1, 2, ..., C\} \to \mathbb{R}$ like for example [3], then we define h_c to be H in which the second input coordinate is fixed to be c. Minimizing the total surrogate risk is not only efficient to estimate posteriors (Section 3): it is also useful to reduce the error in label prediction, as the total surrogate risk is an upperbound for the *Hamming risk* [3]: $\varepsilon_{\mathcal{S}}^{\mathrm{H}}(\mathcal{H}) \doteq (1/(mC)) \sum_{c=1}^{C} \sum_{i=1}^{m} \mathrm{I}[y_{ic} h_c(\boldsymbol{x}_i) < 0]$, where I[.] denotes the indicator variable. It is indeed not hard to check that for any strictly convex surrogate loss ψ, we have $\varepsilon_{\mathcal{S}}^{\mathrm{H}}(\mathcal{H}) \leq (1/\psi(0)) \times \varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H})$. We are left with the following question about UNN:

"are there sufficient conditions on the surrogate loss ψ that guarantee, under the sole **WLA**, a *convergence rate* towards the optimum of (15) with UNN ?"

We give a positive answer to this question when the surrogate loss meets the following smoothness requirement.

Definition 3. *[18] ψ is said to be ω strongly smooth iff there exists some $\omega > 0$ such that, for all $x, x' \in \mathrm{int}(\mathrm{dom}(\psi))$, $D_\psi(x' \| x) \leq \frac{\omega}{2}(x' - x)^2$, where*

$$D_\psi(x' \| x) \doteq \psi(x') - \psi(x) - (x' - x)\nabla_\psi(x) \tag{16}$$

denotes the Bregman divergence with generator ψ [2].

Denote $n_j \doteq |\{i : j \sim_k i\}|$ the number of examples in $\mathcal{S}$ of which $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ is a nearest neighbor, and $n_* \doteq \max_j n_j$. Denote also $\mathcal{H}_{\mathrm{opt}}$ the leveraged k-NN which minimizes $\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H})$; it corresponds to the set of classifiers $\hat{h}_{\mathrm{opt}}$ of Section 3 that would minimize (3) over each class. We are now ready to state our main result (remark that $\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H}_{\mathrm{opt}}) \leq \psi(0)$).

Theorem 3. *Suppose (**WLA**) holds and choose as ψ is any ω strongly smooth, strictly convex loss. Then for any fixed $\tau \in [\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H}_{\mathrm{opt}}), \psi(0)]$, UNN has fit a leveraged k-NN classifier $\mathcal{H}$ satisfying $\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H}) \leq \tau$ provided the number of boosting iterations T in the inner loop satisfies:*

$$T \geq \frac{(\psi(0) - \tau)\omega m n_*}{2\vartheta^2 \varrho^2} \ . \tag{17}$$

Proof sketch: To fit UNN to the notations of (15), we let h_c represent the leveraged k-NN in which each $\boldsymbol{\alpha}_j$ is restricted to α_{jc}. We first analyze $\varepsilon_{\mathcal{S}}^{\psi}(h_c, c)$ for some fixed c in the outer loop of Algorithm 1, after all α_{jc} have been computed in the inner loop. We adopt the following notations in this proof: we plug in the weight notation the iteration t and class c, so that $w_{ti}^{(c)}$ denotes the weight of example $\boldsymbol{x}_i$ at the beginning of the "**for** c" loop of Algorithm 1.

ψ is ω strongly smooth is equivalent to $\tilde{\psi}$ being strongly convex with parameter ω^{-1} [18], that is,

$$\tilde{\psi}(w) - \frac{1}{2\omega} w^2 \text{ is convex}, \tag{18}$$

where we use notation $\tilde{\psi}(x) \doteq \psi^{\star}(-x)$. Any convex function h satisfies $h(w') \geq h(w) + \nabla_h(w)(w' - w)$. We apply this inequality taking as h the function in (18). We obtain, $\forall t = 1, 2, ..., T, \forall i = 1, 2, ..., m, \forall c = 1, 2, ..., C$:

$$D_{\tilde{\psi}}\left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti}\right) \geq \frac{1}{2\omega}\left(w^{(c)}_{(t+1)i} - w^{(c)}_{ti}\right)^2 \quad . \tag{19}$$

On the other hand, Cauchy-Schwartz inequality and (12) yield:

$$\forall j \in \mathcal{S}, \sum_{i:j\sim_k i}\left(\mathrm{r}^{(c)}_{ij}\right)^2 \sum_{i:j\sim_k i}(w^{(c)}_{(t+1)i} - w^{(c)}_{ti})^2 \geq \left(\sum_{i:j\sim_k i}\mathrm{r}^{(c)}_{ij} w^{(c)}_{ti}\right)^2 \quad . \tag{20}$$

Lemma 1. *Under the **WLA**, index j returned by* WIC *at iteration t satisfies* $\left|\sum_{i:j\sim_k i} w^{(c)}_{ti}\mathrm{r}^{(c)}_{ij}\right| \geq 2\vartheta\varrho$.

(proof omitted) Letting $e(t) \in \{1, 2, ..., m\}$ denote the index of the example returned at iteration t by WIC in Algorithm 1, and using (19), (20), Lemma 1 in this order, we arrive after few derivations (skipped because of the lack of space):

$$\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti}\right) \geq \frac{1}{2\omega m}\sum_{i:e(t)\sim_k i}\left(w^{(c)}_{(t+1)i} - w^{(c)}_{ti}\right)^2 \geq \frac{2\vartheta^2\varrho^2}{\omega m n_{e(t)}} \geq \frac{2\vartheta^2\varrho^2}{\omega m n_*} \quad .$$

Summing these inequalities for $t = 1, 2, ..., T$ yields:

$$\sum_{t=1}^{T}\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti}\right) \geq \frac{2T\vartheta^2\varrho^2}{\omega m n_*} \quad . \tag{21}$$

Now, UNN meets the following property ([8], A.2):

$$\varepsilon^{\psi}_{\mathcal{S}}(h_{(t+1)c}, c) - \varepsilon^{\psi}_{\mathcal{S}}(h_{tc}, c) = -\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti}\right), \tag{22}$$

where $h_{(t+1)c}$ denotes h_c after the t^{th} iteration in the inner loop of Algorithm 1. We unravel (22), using the fact that all $\boldsymbol{\alpha}$ are initialized to the null vector, and obtain that at the end of the inner loop, h_c satisfies:

$$\varepsilon^{\psi}_{\mathcal{S}}(h_c, c) = \psi(0) - \sum_{t=1}^{T}\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti}\right) \leq \psi(0) - \frac{2T\vartheta^2\varrho^2}{\omega m n_*} \quad , \tag{23}$$

from (21). There remains to compute the minimal value of T for which the right hand side of (23) becomes no greater than some user-fixed $\tau \in [0, 1]$ to obtain that $\varepsilon^{\psi}_{\mathcal{S}}(h_c, c) \leq \tau$. The aggregation of the bounds for each $c = 1, 2, ..., C$ in $\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H})$ is immediate as it is an average of $\varepsilon^{\psi}_{\mathcal{S}}(h_c, c)$ over all classes. Hence, this minimal value of T, used for each $c = 1, 2, ..., C$, also yields $\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}) \leq \tau$. This ends the proof of Theorem 3. □

Table 2. Computation of δ_{jc} and the weight update rule of our implementation of UNN, for the strictly convex losses of Table 1. UNN leverages example j for class c, and the weight update is that of example i (See text for details and notations).

	δ_{jc}, see (26)	$g : w_i \leftarrow g(w_i)$
A	$2W_{jc} - 1$	$w_i - 2\delta_{jc} y_{ic} y_{jc}$
B	$\ln \frac{W_{jc}}{1-W_{jc}}$	$\frac{w_i}{w_i \ln 2 + (1 - w_i \ln 2) \times \exp(\delta_{jc} y_{ic} y_{jc})}$
C	$\log_2 \frac{W_{jc}}{1-W_{jc}}$	$\frac{w_i}{w_i + (1-w_i) \times 2^{\delta_{jc} y_{ic} y_{jc}}}$
D	$\frac{2W_{jc}-1}{2\sqrt{W_{jc}(1-W_{jc})}}$	$1 - \frac{1 - w_i + \sqrt{w_i(2-w_i)}\delta_{jc} y_{ic} y_{jc}}{\sqrt{1 + \delta_{jc}^2 w_i (2-w_i) + 2(1-w_i)\sqrt{w_i(2-w_i)}\delta_{jc} y_{ic} y_{jc}}}$
E	N/A	N/A
F	$\frac{1}{2} \ln \frac{W_{jc}}{1-W_{jc}}$	$\exp(-\delta_{jc} y_{ic} y_{jc})$
G	$\frac{4}{1-\alpha^2} \left(\frac{(W_{jc})^{\frac{2}{1-\alpha}} - (1-W_{jc})^{\frac{2}{1-\alpha}}}{(W_{jc})^{\frac{2}{1-\alpha}} + (1-W_{jc})^{\frac{2}{1-\alpha}}} \right)$	$\frac{4}{1-\alpha^2} \times \left(\frac{1-\alpha^2}{4} \delta_{jc} y_{ic} y_{jc} + \left(\frac{1+\alpha}{2\sqrt{w_i}} \right)^{1-\alpha} \right)^{-\frac{2}{1-\alpha}}$

Section 3 has underlined the importance of balanced convex losses in obtaining simple efficient estimators for conditional class probabilities. Coupled with Theorem 3, we now show that UNN may be a fast approach to obtain such estimators.

Corollary 1. *Consider any permissible ϕ that has been scaled without loss of generality so that $\phi'(1/2) = 1$, $\phi(0) = \phi(1) = 0$. Then for the corresponding balanced convex loss $\psi = \psi_\phi$ and under the **WLA**, picking*

$$T > \frac{mn_*}{2\vartheta^2 \varrho^2 \min_{x \in (0,1)} \left| \frac{\partial^2 \phi}{\partial x^2} \right|} \tag{24}$$

in the inner loop of UNN, for each $c = 1, 2, ..., C$, guarantees to yield an optimal leveraged k-NN $\mathcal{H}$, satisfying $\varepsilon_{\mathrm{S}}^{\psi}(\mathcal{H}) = \varepsilon_{\mathrm{S}}^{\psi}(\mathcal{H}_{\mathrm{opt}})$. This leveraged k-NN yields efficient estimators for conditional class probabilities, for each class, by computing:

$$\hat{p}_c(\boldsymbol{x}) = \nabla_{\phi}^{-1}(h_c(\boldsymbol{x})) \ . \tag{25}$$

(Proof omitted) For the most popular permissible functions (Table 1), quantity $\min_{x \in (0,1)} \left| \frac{\partial^2 \phi}{\partial x^2} \right|$ does not take too small value: its values are respectively 8, $4/\ln 2$, 4 for the permissible functions corresponding to the squared loss, logistic loss, Matsushita loss. Hence, in these cases, the bound for T in (24) is not significantly affected by this term.

5 Experiments

5.1 Computing Leveraging Coefficients and Weights Update

Fix for short $\mathcal{S}_{jb}^{(c)} \doteq \{i : j \sim_k i \wedge y_{ic} = b y_{jc}\}$ for $b \in \{+, -\}$. (12) may be simplified as $\sum_{i \in \mathcal{S}_{j+}^{(c)}} \nabla_\psi \left(\delta + \nabla_\psi^{-1}(-w_i) \right) = \sum_{i \in \mathcal{S}_{j-}^{(c)}} \nabla_\psi \left(-\delta + \nabla_\psi^{-1}(-w_i) \right)$. There

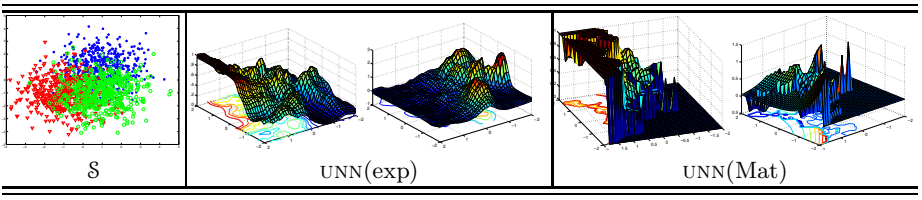


Fig. 1. From left to right: example of simulated dataset with $\sigma = 1.1$; the estimated posterior for class 1 obtained by UNN(exp); the corresponding gridwise KL divergence for class 1; the estimated posterior for class 1 obtained by UNN(Mat); the corresponding gridwise KL divergence for class 1 (see (28) and text for details).

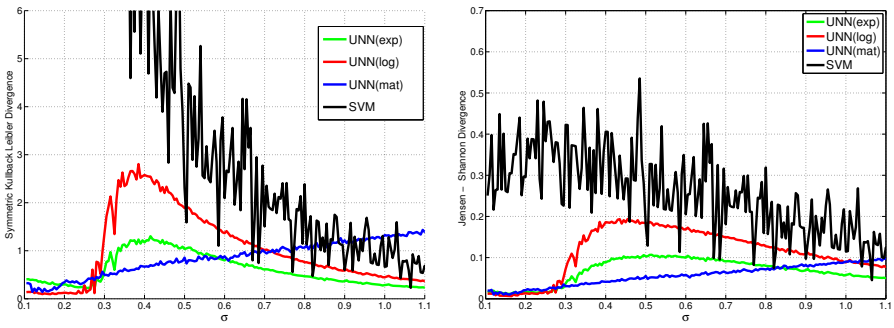


Fig. 2. Average Symmetric KL-divergence (left) and JensenShannon divergence (right) as a function of σ on simulated datasets, for UNN(exp), UNN(log), UNN(Mat) (left, $k = 10$) and SVM .

is no closed form solution to this equation in the general case. While it can be simply approximated with dichotomic search, it buys significant computation time, as this approximation has to be performed for each couple (c, t). We tested a much faster alternative which produces results that are in general experimentally quite competitive, consisting in solving instead: $\sum_{i \in \mathcal{S}_{j+}^{(c)}} w_i \nabla_\psi(\delta) = \sum_{i \in \mathcal{S}_{j-}^{(c)}} w_i \nabla_\psi(-\delta)$. We get equivalently that δ satisfies:

$$\frac{\nabla_\psi(-\delta)}{\nabla_\psi(\delta)} = \frac{W_{jc}}{1 - W_{jc}} \ , \tag{26}$$

with $W_{jc} \doteq (\sum_{i \in \mathcal{S}_{j+}^{(c)}} w_i)/(\sum_{i \in \mathcal{S}_{j+}^{(c)}} w_i + \sum_{i \in \mathcal{S}_{j-}^{(c)}} w_i)$. Remark the similarity with (8). Table 2 gives the corresponding expressions for δ and the weight updates.

5.2 General Experimental Settings

We have tested three flavors of UNN: with the exponential loss (F in Table 1), the logistic loss (B in Table 1) and Matsushita's loss (D in Table 1). All three

a respectively referred to as UNN(exp), UNN(log) and UNN(Mat). It is the first time this last flavor is tested, even from the classification standpoint. We chose support vector machines (SVM) as the contender against which to compare UNN: SVM are large margin classifiers with convenient methods to obtain estimators for the posteriors [19]. For all these algorithms, we compute the estimation of posteriors as follows: we use (11) for UNN(exp), (25) for UNN(log) and UNN(Mat). For SVM, we use the method of [19], which, given a SVM f for class c, forms the posterior:

$$\hat{p}_c(\boldsymbol{x}) \doteq \frac{1}{1+\exp(af(\boldsymbol{x})+b)} \ , \tag{27}$$

where a and b are estimated by maximizing the log-likelihood of the training sample with a five-fold cross validation. On synthetic datasets SVM performs equally using both linear and radial basis function kernel. Therefore, in the following we indicate with SVM the linear Support Vector Machine. We use three metrics to evaluate the algorithms: Two computed to evaluate performance on synthetic data and one to evaluate performance on real datasets. On simulated data, we compute first Kullback-Leibler (KL) divergences between the true and estimated posterior and after their mean obtaining the Symmetric Kullback-Leibler divergences:

$$D_{\mathrm{KL}}(\hat{p}\|p) \doteq \sum_c \mathbf{Pr}[c] \int \mathbf{Pr}[\boldsymbol{x}]\hat{p}_c(\boldsymbol{x}) \ln \frac{\hat{p}_c(\boldsymbol{x})}{p_c(\boldsymbol{x})} \mathrm{d}\mu \ , \ D_{\mathrm{KL}}(p\|\hat{p}) \doteq \sum_c \mathbf{Pr}[c] \int \mathbf{Pr}[\boldsymbol{x}] \ p_c(\boldsymbol{x}) \ln \frac{p_c(\boldsymbol{x})}{\hat{p}_c(\boldsymbol{x})} \mathrm{d}\mu \tag{28}$$

$$SymmD_{\mathrm{KL}} \doteq \frac{1}{2}(D_{\mathrm{KL}}(\hat{p}\|p) + D_{\mathrm{KL}}(p\|\hat{p})) \tag{29}$$

and also we compute JensenShannon (JS) divergence:

$$D_{\mathrm{JS}} \doteq \frac{1}{2}(D_{\mathrm{KL}}(\hat{p}\|q) + D_{\mathrm{KL}}(p\|q)) \tag{30}$$

where q is the average of the two distribution. Our estimate, $Symm\hat{D}_{\mathrm{KL}}$ and $\hat{D_{\mathrm{JS}}}$ rely on a simple fine-grained grid approximation of the integral over the subsets of $\mathcal{O}$ of sufficient mass according to μ. On real data, we compute a couple of metrics. First, we compute the F-measure of the classifiers (the harmonic average of precision and recall), based on thresholding the probabilistic output and deciding that $\boldsymbol{x}$ belong to class c iff $\hat{p}_c(\boldsymbol{x}) \geq \kappa$, for varying $\kappa \in (1/2, 1)$. Second, we compute the rejection rate, that is, the proportion of observations for which $\hat{p}_c(\boldsymbol{x}) < \kappa$. Either we plot couples of curves for the F-measure and rejection rates, or we summarize both metrics by their average values as κ ranges through $(1/2, 1)$, which amounts to compute the area under the corresponding curves.

5.3 Results on Simulated Data

We evaluated the goodness-of-fit of the estimates on simulated datasets with the following experiments. We crafted a general domain consisting of $C = 3$

Table 3. Average results over simulated data, for UNN(exp), UNN(log), UNN(Mat) with four different values of k, and for support vector machines with linear (SVM)

	k	UNN(exp)	UNN(log)	UNN(Mat)	SVM
$Symm\hat{D}_{\mathrm{KL}}$	10	0.599	1.029	0.848	3.533
	20	0.372	0.760	0.687	
	30	0.293	0.610	0.646	
	40	0.254	0.534	0.632	
$\hat{D}_{\mathrm{JS}}$	10	0.067	0.113	0.0562	0.256
	20	0.045	0.086	0.045	
	30	0.036	0.072	0.043	
	40	0.032	0.065	0.043	
F-measure	10	90.32	89.59	90.58	91.02
	20	90.62	89.53	90.81	
	30	90.70	89.26	90.84	
	40	90.72	88.82	90.88	

equiprobable classes, each of which follows a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \sigma \mathrm{I})$, for $\sigma \in [0.1, 1.1]$ with steps of 0.005, and $\boldsymbol{\mu}$ remains the same. For each value of σ, we compute the average over ten simulations, each of which consists of 1500 training examples and 4500 testing examples. We get overall several thousands datasets, on which all algorithms are tested. Figure 1 presents an example of such datasets, along with results obtained by UNN(exp) and UNN(Mat) from the standpoints of the posterior estimates and KL-divergence on the same class. The estimators are rather good, with the largest mismatches (KL-divergence) located near the frontiers of classes. Also, UNN(Mat) tends to outperform UNN(exp).

Figure 2 synthesizes the results from the KL and JS divergence standpoints. Two clear conclusions can be drawn from these results. First, UNN is the clear winner over SVM for the posteriors estimation task. The results of each flavor of UNN is indeed better than those of SVM by orders of magnitude. This is all the more important as the kernels we used are the theoretical kernels of choice given the way we have simulated data. The second conclusion is that UNN(Mat) is the best of all flavors of UNN, a fact also confirmed by the synthetic results of Table 3. Its behavior (Figure 2) is also monotonous: it is predictable that it increases with the degree of overlap between classes, that is, with σ. From the *classification* standpoint, the average F-measure metrics display a very slight advantage to SVM.

The most important conclusion that can be drawn from the simulated data is shown in Figure 3: as the number of boosting iterations T increase, UNN does *not* overfit posteriors in general. The only hitch — not statistically significant — is the case $\sigma > 0.7$ for UNN(Mat), but the differences are of very small order compared to the standard deviations of the KL-divergence.

5.4 Results on the SUN Database Domains

We have crafted, out of the challenging SUN computer vision database [20], three datasets, consisting in taking all pictures from the first ten (SUN 10), twenty (SUN 20) or thirty (SUN 30) classes. We have compared UNN(exp), UNN(log), UNN(Mat) and SVM on each dataset, by computing the average values, over

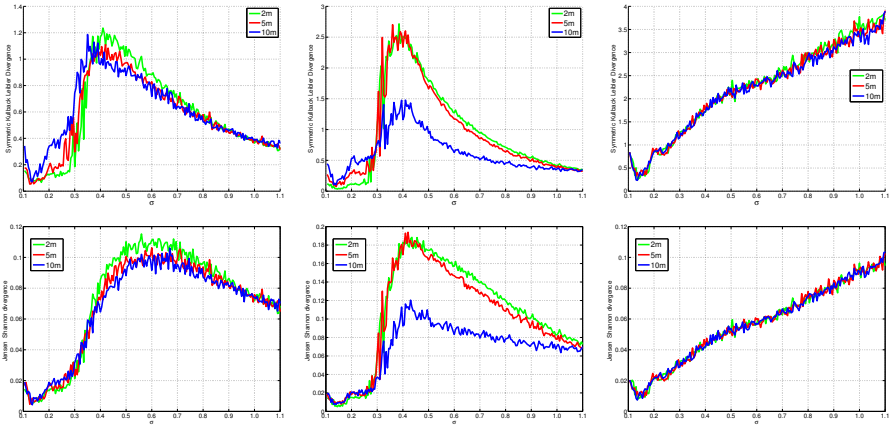


Fig. 3. Average symmetric KL-divergence (top) and JensenShannon divergence (bottom) as a function of σ on simulated datasets, for UNN(exp) (left), UNN(log) (center), UNN(Mat) (right), when the number of boosting iterations T varies in $\{2m, 5m, 10m\}$. The color code in the same on each plot.

Table 4. Area under the (F)-measure (in percentage) and (R)ejection rate on the SUN databases. For each database, the best F and R are written in **bold faces**.

	UNN(exp)		UNN(log)		UNN(Mat)		SVM_l	
	F	R	F	R	F	R	F	R
SUN 10	**89.91**	21.35	84.46	5.18	72.47	**3.39**	87.99	22.32
SUN 20	**82.82**	36.64	72.34	8.51	55.46	**2.51**	74.60	33.25
SUN 30	**73.39**	49.92	61.02	14.99	40.83	**5.99**	62.81	39.95

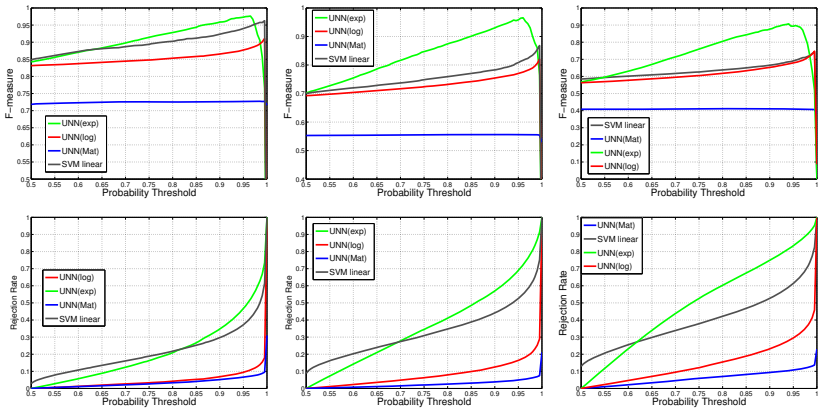


Fig. 4. F-measure (top row) and rejection rates (bottom row) on the SUN domains, with $C = 10$ (left), $C = 20$ (center) and $C = 30$ (right, see Table 3 for notations)

the threshold κ, of the F-measure and the rejection rate. Table 4 summarizes the results obtained. This table somehow confirms that classification and posterior estimation may be conflicting goals when it comes to boosting [4, 5], as UNN(Mat) achieves very poor results compared to the other algorithms. Furthermore, UNN(exp) appears to the clear winner over all algorithms for this classification task. These results have to be appreciated in the light of the rejection rates: in comparison with the other algorithms, UNN(Mat) rejects a very small proportion of the examples, this indicating a high recall for the algorithm. Figure 4 completes the picture by detailing F-measure and rejection rates plots. The F-measure plots clearly display the better performances of UNN(exp) compared to the other algorithms, and the fact that UNN(Mat) displays very stable performances. The rejection rates plots show that UNN(Mat) indeed rejects a very small proportion of examples, even for large values of κ.

6 Conclusion

Boosting algorithms are remarkably simple and efficient from the classification standpoint, and are being used in a rapidly increasing number of domains and problems [6]. In some sense, it would be too bad that such successes be impeded when it comes to posterior estimation [5]. Experimental results display that this estimation is possible, but it necessitates a very fine tuning of the algorithms [6]. The point of our paper is that estimating class conditional probabilities may be possible, without such tedious tunings, and sometimes even *without overfitting*, if we boost topological approaches to learning like nearest neighbors [8]; such approaches offer very promising results in the field of computer vision [7]. There is a simple explanation to this fact. For any classifier, the conditional class probability estimation for some $\boldsymbol{x}$ in (7) is be the same as for any other observation in the vicinity of $\boldsymbol{x}$, where the "vicinity" is to be understood from the *classifier* standpoint. When boosting decision trees, the vicinity of $\boldsymbol{x}$ corresponds to observations classified by the same leaf as $\boldsymbol{x}$. As the number of leaves of the tree increases, the vicinity gets narrowed, which weakens the estimation in (7) and thus overfits the corresponding estimated density. Ultimately, linear combinations of such trees, such as those performed in AdaBoost, make such a fine-grained approximation of the local topology of data that the estimators get irreparably confined to the borders of the interval $[0, 1]$ [5]. Nearest neighbors do not have such a drawback, as the set of k-nearest neighbors in $\mathcal{S}$ of some observation $\boldsymbol{x}$ spans a region of $\mathcal{O}$ which does not change throughout the iterations. Furthermore, nearest neighbor rules exploit a topology of data which, under regularity conditions about the true posteriors, also carries out information about these posteriors. For these reasons, nearest neighbors might be a key entry for a reliable estimation of posteriors with boosting.

Acknowledgments. R. Nock acknowledges a visiting grant from Institut Universitaire de France / Université de Nice.

References

1. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: a new explanation for the effectiveness of voting methods. Annals of Statistics 26, 1651–1686 (1998)
2. Nock, R., Nielsen, F.: On the efficient minimization of classification-calibrated surrogates. In: NIPS*21, pp. 1201–1208 (2008)
3. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning Journal 37, 297–336 (1999)
4. Friedman, J., Hastie, T., Tibshirani, R.: Additive Logistic Regression: a Statistical View of Boosting. Annals of Statistics 28, 337–374 (2000)
5. Buja, A., Mease, D., Wyner, A.-J.: Comment: Boosting algorithms: regularization, prediction and model fitting. Statistical Science 22, 506–512 (2007)
6. Bühlmann, P., Hothorn, T.: Boosting algorithms: regularization, prediction and model fitting. Statistical Science 22, 477–505 (2007)
7. Nock, R., Piro, P., Nielsen, F., Bel Haj Ali, W., Barlaud, M.: Boosting k-NN for categorization of natural scenes. International Journal of Computer Vision (to appear, 2012)
8. Piro, P., Nock, R., Nielsen, F., Barlaud, M.: Leveraging k-NN for generic classification boosting. Neurocomputing 80, 3–9 (2012)
9. Bartlett, P., Jordan, M., McAuliffe, J.D.: Convexity, classification, and risk bounds. Journal of the Am. Stat. Assoc. 101, 138–156 (2006)
10. Nock, R., Nielsen, F.: Bregman divergences and surrogates for learning. IEEE Trans. on Pattern Analysis and Machine Intelligence 31(11), 2048–2059 (2009)
11. Kearns, M.J., Mansour, Y.: On the boosting ability of top-down decision tree learning algorithms. Journal of Comp. Syst. Sci. 58, 109–128 (1999)
12. Amari, S.-I., Nagaoka, H.: Methods of Information Geometry. Oxford University Press (2000)
13. Müller-Funk, U., Pukelsheim, F., Witting, H.: On the attainment of the Cramér-Rao bound in L_r-differentiable families of distributions. Annals of Statistics, 1742–1748 (1989)
14. Cover, T.-M., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. on Information Theory 13, 21–27 (1967)
15. Devroye, L., Györfi, L., Lugosi, G.: A Probabilistic Theory of Pattern Recognition. Springer (1996)
16. Sebban, M., Nock, R., Lallich, S.: Boosting Neighborhood-Based Classifiers. In: Proc. of the 18th International Conference on Machine Learning, pp. 505–512. Morgan Kaufmann (2001)
17. Sebban, M., Nock, R., Lallich, S.: Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problems. J. of Mach. Learn. Res. 3, 863–885 (2003)
18. Kakade, S., Shalev-Shwartz, S., Tewari, A.: Applications of strong convexity–strong smoothness duality to learning with matrices. Technical Report CoRR abs/0910.0610, Computing Res. Repository (2009)
19. 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)
20. Xiao, J., Hays, J., Ehringer, K.-A., Oliva, A., Torralba, A.: SUN database: Large-scale scene recognition from abbey to zoo. In: Proc. of IEEE International Conference on Computer Vision and Pattern Recognition, pp. 3485–3492 (2010)

Diversity Regularized Ensemble Pruning

Nan Li[1,2], Yang Yu[1], and Zhi-Hua Zhou[1]

[1] National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210046, China
[2] School of Mathematical Sciences, Soochow University, Suzhou 215006, China
{lin,yuy,zhouzh}@lamda.nju.edu.cn

Abstract. Diversity among individual classifiers is recognized to play a key role in ensemble, however, few theoretical properties are known for classification. In this paper, by focusing on the popular ensemble pruning setting (i.e., combining classifier by *voting* and measuring diversity in *pairwise* manner), we present a theoretical study on the effect of diversity on the generalization performance of voting in the PAC-learning framework. It is disclosed that the diversity is closely-related to the hypothesis space complexity, and encouraging diversity can be regarded to apply regularization on ensemble methods. Guided by this analysis, we apply explicit diversity regularization to ensemble pruning, and propose the *Diversity Regularized Ensemble Pruning* (DREP) method. Experimental results show the effectiveness of DREP.

Keywords: diversity, ensemble pruning, diversity regularization.

1 Introduction

Ensemble methods [33], which train multiple classifiers for one single task, are among the state-of-the-art machine learning approaches. It is widely accepted that ensemble methods usually achieve better generalization performance than single classifiers, and they have achieved great successes in a large variety of real-word applications.

Generally speaking, an ensemble is built in two steps: first multiple classifiers are trained for one task, and then these classifiers are combined together to get a better performance in some manners like voting. Given multiple trained individual classifiers, instead of combining all of them, there are many studies tring to select a subset from them to comprise the ensemble [28]. In the literature, the task of reducing ensemble sizes is called as ensemble pruning [19], selective ensemble [35], ensemble selection [6] or ensemble thinning [1]. Currently, we do not distinguish between them and use ensemble pruning for simplicity. By producing ensembles of smaller sizes, ensemble pruning has the apparent advantage of improving storage and computational efficiency for predictions. Furthermore, both theoretical and empirical studies have shown that ensemble pruning can also improve the generalization performance of ensemble [35,6,32,20], that is, the pruned ensemble can achieve better performance than the complete ensemble.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 330–345, 2012.

In the ensemble pruning literature, greedy pruning methods which search the space of possible classifier subsets by taking greedy local search have drawn much attention [24,20], it is because compared with methods that directly select optimal or near-optimal classifier subsets, they are able to achieve comparative performance and robustness at much smaller computational costs. There are two salient parameters in greedy pruning methods: the direction for searching the space (i.e., forward and backward) and the criterion for evaluating available actions at each search step. Since it is shown that the direction does not significantly affect the performance [24], much attention has been paid on the design of the evaluation criterion. As diversity among individual classifiers is widely recognized to be key to the success of an ensemble, many evaluation criteria have been developed to select diverse individual classifiers, mainly by smart heuristics [1,23,20,24]. In practice, although positive correlation has been demonstrated between diversity and accuracy of ensemble [9,16,11], few theoretical prosperities of ensemble diversity is known. Moreover, the usefulness of exploiting diversity measures in building stronger ensemble was doubted in [15,27].

In this paper, concentrating on a popular setting of ensemble pruning where the individual classifiers are combined by *voting* and the diversity is measured in the *pairwise* manner, we present a theoretical analysis on the effect of diversity on the generalization performance of voting based on the probably approximately correct learning (PAC-learning) framework [29]. To our best knowledge, this is the first PAC-style analysis on diversity's effect on voting. We show that encouraging larger diversity leads to smaller hypothesis space complexity and thus better generalization performance, which implies that controlling diversity can be regarded to apply regularization on ensemble methods. Then, guided by the theoretical analysis, we propose the DREP method which is a greedy forward ensemble pruning method with explicit diversity regularization. Experimental results show the effectiveness of the DREP method.

The remainder of the paper is organized as follows. Section 2 gives a brief review on ensemble selection and ensemble diversity. Section 3 presents our theoretical study on the role of diversity in voting. Based on the theoretical results, Section 4 proposes the DREP method, followed by Section 5 which reports on the experimental results. Finally, the paper is concluded in Section 6.

2 Related Work

With the goal of improving storage and computational efficiency as well as generalization performance, ensemble pruning deals with the problem of reducing ensemble sizes. The first work on this topic was possibly done by Margineantu and Dietterich [19], which tried to prune AdaBoost, but later Tamon and Xiang [26] showed that the boosting pruning problem is intractable even to approximate. Instead of pruning ensembles generated by sequential methods, Zhou et al. [35] and Caruana et al. [6] respectively studied on pruning ensembles generated by parallel methods such as Bagging [3] and parallel heterogeneous ensembles consisting of different types of individual classifiers, and it was shown that better

performance can be obtained at smaller ensemble sizes. Ever since, most ensemble pruning studies were devoted to parallel ensemble methods.

Given a set of trained classifiers, selecting the sub-ensemble with the best generalization performance is difficult mainly due to two reasons: First, it is not easy to estimate the generalization performance of a sub-ensemble; second, finding the optimal subset is a combinatorial search problem with exponential computational complexity, thus it is unfeasible to compute the exact solution by exhaustive search and approximate search is needed. In the past decade, a number of methods have been proposed to overcome this difficulty [28], which can be roughly classified into two groups based on their employed search methods. The first group of methods use *global search* to directly select the optimal or near-optimal classifier subset. In the literature, many techniques have been used, such as genetic algorithm [35], semi-definite programming [31], clustering [12,17], sparse optimization with sparsity-inducing prior [7] or ℓ_1-norm constraint [18], etc. In practice, this kind of methods can achieve good performance, but their computational costs are usually quite large.

The second group of ensemble pruning methods is based on *greedy local search* of the space of all possible ensemble subsets [20,24]. According to the search direction, this group of methods can be further divided into greedy *forward* pruning methods which start with empty set and iteratively add the classifier optimizing certain criterion, and greedy *backward* methods that start with the complete ensemble and iteratively eliminate classifiers. It has been shown that greedy pruning methods are able to achieve comparative performance and robustness with global search methods but at much smaller computational costs [20,13]. Moreover, based on extensive experiments, Partalas et al. [24] suggested to use the greedy forward methods because both directions achieve similar performance but the forward direction produces smaller ensemble sizes. Then, the study of greedy pruning methods was mainly devoted to the criterion that is used for evaluating available actions at each local search step. Since the diversity within an ensemble is widely recognized to be important to its success, many criteria have been proposed to select diverse individual classifiers, such as Kappa [19,1], complementarity [21,20], orientation [22,20], margin distance [21,20], FES [23], etc. It is easy to see that most of these criteria are based on smart heuristics.

In practice, the importance of diversity was first discovered from error analysis for regression [14], and then extended to classification. For classification, it has been observed from empirical studies like [9] that there exists positive correlation between diversity and accuracy of ensemble. Also, some theoretical studies have shown that encouraging diversity is beneficial. For example, Kuncheva et al. [16] found that negative dependence between individual classifiers is beneficial to the accuracy of an ensemble, Fumera and Roli [11] found that the performance of ensemble depends on the performance of individual classifiers and their correlation. Based on current results, we can see that it is no problem to reach that encouraging diversity is beneficial to the performance of ensemble, but it is hard to tell the theoretical properties of diversity in ensemble. In the famous margin explanation of voting [25], the diversity is totally not considered in the framework.

Also, some doubts have been raised on the usefulness of exploiting diversity measures in building stronger ensembles [15,27]. Therefore, understanding ensemble diversity remains an important issue in ensemble learning and further investigations are needed. Recently, by defining diversity and ensemble combination rule in the parameter space, our previous work [30] showed that diversity control can play a role of regularization as in statistical learning methods, which, however, relies on linear classifiers and average combination and thus cannot be applied to other kind of classifiers and voting combination. In this work, we consider the popular ensemble pruning setting, i.e., the diversity is measured in the output space which leaves the specification of classifiers unimportant, and the individual classifiers are combined by voting.

3 Diversity and Generalization Performance of Voting

In ensemble pruning, voting is one of the most widely used methods to combine individual classifiers. In this section, we give a theoretical study on the effect of diversity on the generalization performance of voting.

3.1 Basics and Diversity Measure

Consider binary classification, given a set of n trained classifiers $H = \{h_i(\boldsymbol{x})\}_{i=1}^n$, where each classifier $h_i : \mathcal{X} \mapsto \{-1, +1\}$ is a mapping from the feature space $\mathcal{X}$ to the class label set $\{-1, +1\}$, the voting rule defines a decision function by taking an average of classifiers in H as

$$f(\boldsymbol{x}; H) = \frac{1}{n} \sum\nolimits_{i=1}^{n} h_i(\boldsymbol{x}) \ , \tag{1}$$

and it predicts the class label of $\boldsymbol{x}$ as $\texttt{sign}[f(\boldsymbol{x}; H)]$. Obviously, it makes wrong prediction on example $(\boldsymbol{x}, y)$ only if $yf(\boldsymbol{x}; H) \leq 0$, and $yf(\boldsymbol{x}; H)$ is called the *margin* of f at $(\boldsymbol{x}, y)$.

Let $\mathcal{D}$ is the underlying distribution over $\mathcal{X} \times \{-1, +1\}$, and $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ is a set of examples randomly sampled from $\mathcal{D}$, the *generalization error* (denoted as $err_g(f)$) and the *empirical error* with margin θ on S (denoted as $err_S^\theta(f)$) are respectively defined as

$$err_g(f) = P_{(\boldsymbol{x},y)\sim\mathcal{D}}[yf(\boldsymbol{x}) \leq 0] \quad \text{and} \quad err_S^\theta(f) = \frac{1}{m} \sum\nolimits_{i=1}^{m} \mathbb{I}[y_i f(\boldsymbol{x}_i) \leq \theta] \ , \tag{2}$$

where $\mathbb{I}[z]$ is the indicator function which takes 1 if z is $\texttt{true}$, and 0 otherwise.

Although there is no generally accepted formal definition of diversity in the literature [5,34], popular diversity measures are usually formalized based on pairwise difference between every pair of individual classifiers [15], such as Q-statistics, correlation coefficient, disagreement measure and κ-statistics. In this work, we also measure diversity based on pairwise difference, and the definition is given as follows.

Definition 1. *Given a set of m examples $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, the diversity of classifier set $H = \{h_i(\boldsymbol{x})\}_{i=1}^n$ on S is defined as*

$$\mathtt{div}(H) = 1 - \frac{1}{\sum_{1 \le i \ne j \le n} 1} \sum\nolimits_{1 \le i \ne j \le n} \mathtt{diff}(h_i, h_j) \ , \tag{3}$$

where $\mathtt{diff}(\cdot, \cdot)$ measures the pairwise difference between two classifiers as

$$\mathtt{diff}(h_i, h_j) = \frac{1}{m} \sum\nolimits_{k=1}^m h_i(\boldsymbol{x}_k) h_j(\boldsymbol{x}_k) \ . \tag{4}$$

It is obvious that the difference $\mathtt{diff}(h_i, h_j)$ falls into the interval $[-1, 1]$, and $\mathtt{diff}(h_i, h_j)$ equals to 1 (or -1) only if two classifiers h_i and h_j always make the same (or opposite) predictions on the data sample S, and the smaller $\mathtt{diff}(h_i, h_j)$, the larger difference between h_i and h_j. Consequently, since the diversity is based on the average of pairwise differences, we can see that the larger $\mathtt{div}(H)$ the larger the diversity of the classifier set H.

It is easy to find that this diversity measure is closely-related with the disagreement measure [15]. Moreover, different from [30] which defines the diversity in the parameter space of classifiers, here this diversity measure is defined in the output space, thus can cover various kinds of individual classifiers.

3.2 Theoretical Results

Our analysis is based on the PAC-learning framework [29], which is one of the most widely used framework for analyzing learning algorithms. Before giving the main results, we first introduce some necessary background.

In learning theory, it is known that the generalization error of a learning algorithm can be bounded by its empirical error and the complexity of feasible hypothesis space [29,2]. Since the hypothesis space is uncountable for many learning methods, the hypothesis space complexity is often described by a quantity called *covering number*, which is defined as below.

Definition 2. *Let B be a metric space with metric ρ. Given a set of m examples $S = \{\boldsymbol{x}_i\}_{i=1}^m$ and a function space $\mathcal{F}$, characterize every $f \in \mathcal{F}$ with a vector $\boldsymbol{v}_S(f) = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_m)]^\top \in B^m$. The covering number in p-norm $\mathcal{N}_p(\mathcal{F}, \epsilon, S)$ is the minimum number l of vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l \in B^m$ such that, for all $f \in \mathcal{F}$ there exists $j \in \{1, \ldots, l\}$,*

$$\|\rho(\mathbf{v}_S(f), \mathbf{u}_j)\|_p = \left(\sum\nolimits_{i=1}^m \rho(f(\boldsymbol{x}_i), u_{j,i})^p \right)^{1/p} \le m^{1/p} \epsilon \ ,$$

and $\mathcal{N}_p(\mathcal{F}, \epsilon, m) = \sup_{S:|S|=m} \mathcal{N}_p(\mathcal{F}, \epsilon, S)$.

Currently, we show how the ensemble diversity affects the generalization performance of voting. In particular, our study mainly focuses on the effect of diversity on the hypothesis space complexity of voting. Before presenting the main result, we give the following lemma.

Lemma 1. *Given a set of classifiers* $H = \{h_i(\boldsymbol{x})\}_{i=1}^n$ *and a set of examples* $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, *denote* $\mathbf{f} = [f(\boldsymbol{x}_1; H), \ldots, f(\boldsymbol{x}_m; H)]^\top$ *be the output of the decision function* f*'s outputs on* S. *On data set* S, *if* $\mathtt{div}(H) \geq q$, *then it follows*

$$\|\mathbf{f}\|_1 \leq m\sqrt{1/n + (1-1/n)(1-q)} \ .$$

Proof. By basic algebra, we have

$$\begin{aligned}\|\mathbf{f}\|_2^2 &= \sum\nolimits_{i=1}^m \left(\frac{1}{n}\sum\nolimits_{t=1}^n h_t(\boldsymbol{x}_i)\right)^2 = \sum\nolimits_{i=1}^m \left(\frac{1}{n} + \frac{1}{n^2}\sum\nolimits_{1\leq j\neq k\leq n} h_j(\boldsymbol{x}_i)h_k(\boldsymbol{x}_i)\right) \\ &= m\left(1/n + (1-\mathtt{div}(H))(1-1/n)\right) \geq 0 \ .\end{aligned}$$

We can find the quantity $1/n + (1-q)(1-1/n)$ is always non-negative. Then, based on the inequality $\|\mathbf{f}\|_1 \leq \sqrt{m}\|\mathbf{f}\|_2$, we can obtain the result directly. □

Theorem 1. *Let* $\mathcal{F}$ *denote the function space such that for every* $f \in \mathcal{F}$, *there exist a set of* n *classifiers* $H = \{h_i(\boldsymbol{x})\}_{i=1}^n$ *satisfying* $f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n h_i(\boldsymbol{x})$ *and* $\mathtt{div}(H) \geq q$ *for any i.i.d. sample* S *of size* m, *then for any* ϵ, *it holds*

$$\log_2 \mathcal{N}_\infty(\mathcal{F}, \epsilon, m) \leq \frac{36(1+\ln n)}{\epsilon^2}\log_2\left(2m\lceil 4\sqrt{1/n + (1-1/n)(1-q)}/\epsilon + 2\rceil + 1\right) \ .$$

Proof. This proof follows similar strategy with Theorem 4 and 5 in [31], here we give the main sketch and focus on the difference. If $\epsilon \geq 1$, the result follows trivially, so it is assumed $\epsilon \leq 1$ subsequently. First, the interval $[-1-\epsilon/2, 1+\epsilon/2]$ is divided into $n = \lceil 4/\epsilon + 2\rceil$ sub-intervals, each of size no larger than $\epsilon/2$, and θ_j be the boundaries of the sub-intervals so that $\theta_j - \theta_{j-1} \leq \epsilon/2$ for all j. Let $j_l(i)$ denote the maximum index of θ_j such that $f(\boldsymbol{x}_i) - \theta_{j_l(i)} \geq \epsilon/2$ and $j_r(i)$ the maximum index of θ_j such that $f(\boldsymbol{x}_i) - \theta_{j_r(i)} \leq -\epsilon/2$. Let

$$\boldsymbol{h}_i = [h_1(\boldsymbol{x}_i), \ldots, h_T(\boldsymbol{x}_i)]^\top, \quad \boldsymbol{h}_i' = [\boldsymbol{h}_i, -\theta_{j_l(i)}]^\top \quad \text{and} \quad \boldsymbol{h}_i'' = [-\boldsymbol{h}_i, \theta_{j_r(i)}]^\top.$$

Then, based on similar steps in [31], the covering number $\mathcal{N}_\infty(\mathcal{F}, \epsilon, S)$ is no more than the number of possible values of the vector $\boldsymbol{\beta}$, which is defined as

$$\boldsymbol{\beta} = g_p\left(\sum\nolimits_{i=1}^m a_i\boldsymbol{h}_i' + \sum\nolimits_{i=1}^m b_i\boldsymbol{h}_i''\right) \ , \tag{5}$$

where $g_p(\boldsymbol{u})$ is a component-wise function mapping each component u_i of $\boldsymbol{u}$ to $p \cdot \mathtt{sign}(u_i)|u_i|^{p-1}$ with $p \geq 2$, and a_i's and b_i's are non-negative integers under the constraint

$$\sum\nolimits_{i=1}^m (a_i + b_i) \leq 36(1+\ln n)/\epsilon^2 \ . \tag{6}$$

It is easy to find that there is an one-to-one mapping between $\boldsymbol{h}_i'$ and $\boldsymbol{h}_i''$, so the number of possible values of $\boldsymbol{h}_i'$ and $\boldsymbol{h}_i''$ equals to that of $\boldsymbol{h}_i'$. Let $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_m)]^\top$ be f's outputs on S, based on Lemma 1, we can obtain that $\|\boldsymbol{f}\|_1 \leq m\sqrt{1/n + (1-1/n)(1-q)}$. Then, based on the definition of $\theta_{j_l(i)}$, we can find that the number of possible values of $\boldsymbol{h}_i'$ is no more than

$$m\lceil 4\sqrt{1/n + (1-1/n)(1-q)}/\epsilon + 2\rceil \ .$$

Consequently, from (5) and (6) we can find that the number of possible values of $(\boldsymbol{\beta}, z)$ is upper-bounded by

$$\left(2m\lceil 4\sqrt{1/n+(1-1/n)(1-q)}/\epsilon+2\rceil+1\right)^{36(1+\ln n)/\epsilon^2},$$

which completes the proof. □

Furthermore, based on Theorem 1 we can obtain the relationship between diversity and generalization performance of voting, which is given as follows.

Corollary 1. *Under the assumptions of Theorem 1, with probability at least* $1-\delta$*, for any* $\theta > 0$*, every function* $f \in \mathcal{F}$ *satisfies the following bound*

$$err_g(f) \leq err_S^{\theta}(f) + \frac{C}{\sqrt{m}}\left(\frac{\ln n \ln\left(m\sqrt{1/n+(1-1/n)(1-q)}\right)}{\theta^2} + \ln\frac{1}{\delta}\right)^{1/2},$$

where C *is a constant.*

Proof. Based on Bartlett's Lemma 4 in [2], we can obtain

$$err_g(f) \leq err_S^{\theta}(f) + \sqrt{\frac{2}{m}\left(\ln \mathcal{N}_{\infty}(\mathcal{F}, \epsilon/2, 2m) + \ln\frac{2}{\delta}\right)}. \tag{7}$$

By applying Theorem 1 on (7), we can obtain the result. □

Above results show that, when other factors are fixed, encouraging high diversity among individual classifiers (i.e., large value of q in Theorem 1 and Corollary 1) will make the hypothesis space complexity of voting small, and thus better generalization performance can be expected.

3.3 Remarks and Discussions

It can be observed from above theoretical analysis that the diversity is directly related to the hypothesis space complexity of voting, and then affects its generalization performance. From the view of statistical learning, controlling ensemble diversity has a direct impact on the size of hypothesis space of voting, indicating that it plays a role similar with regularization as in popular statistical learning methods. In other words, it implies that encouraging diversity can be regarded to apply regularization on ensemble methods. Also, this result show that encouraging diversity is beneficial but not straightforwardly related to the ensemble accuracy, which coincides with previous study in [16].

To our best knowledge, this work provides the first PAC-style analysis on the role of diversity in voting. The margin explanation of voting presented in [25] is also in the PAC-learning framework, but it is obvious that our work is significantly different because diversity is considered explicitly. The hypothesis space complexity of voting becomes small when the diversity increases, but it is simply characterized by the VC-dimension of individual classifier in [25]. Intuitively, due to the diversity, some parts of the hypothesis space of voting are infeasible, excluding these parts leads to tighter bounds, while assuming the hypothesis space compact makes the bounds looser.

4 Diversity Regularized Ensemble Pruning

In this section, we apply above theoretical analysis to ensemble pruning, and propose the *Diversity Regularized Ensemble Pruning* (DREP) method, which is a greedy forward ensemble pruning method.

The main difference between DREP and existing greedy pruning methods lies in the criterion for evaluating available actions at each step. In the previous section, it is shown in Corollary 1 that the generalization performance of an ensemble depends on its empirical error and diversity, so it is natural to design the evaluation criterion accordingly. However, it is easy to see that in the bound the diversity has complicated operations with factors including the sample size m, number of classifier n, η and θ, etc. Then for a given problem, it will be difficult to specify the tradeoff between empirical error and diversity. Hence, a tradeoff parameter is involved in the proposed method.

Moreover, it is easy to see from (3) that when we want to evaluate the diversity of a new ensemble which is obtained by adding one individual classifier, it is needed to compute the pairwise difference between the new added classifier and all the existing classifiers. As a consequence, at each step, if there are many candidate individual classifiers, directly evaluating diversity based on the definition will be of high computational complexity. To avoid this issue, we use a more efficient way based on the following proposition.

Propsition 1. *Given a classifier $h'(\boldsymbol{x})$ and a classifier set $H = \{h_i(\boldsymbol{x})\}_{i=1}^n$, let $H' = H \cup \{h'(\boldsymbol{x})\}$, the diversity of H' on $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ is*

$$\mathtt{div}(H') = \frac{2}{n+1} + \frac{n-1}{n+1}\,\mathtt{div}(H) - \frac{2}{n+1}\,\mathtt{diff}(h', H) \tag{8}$$

where $\mathtt{div}(H)$ is the diversity of H on S and $\mathtt{diff}(h', H)$ measures the difference between new classifier $h'(\boldsymbol{x})$ and H as

$$\mathtt{diff}(h', H) = \frac{1}{m}\sum\nolimits_{i=1}^{m} h'(\boldsymbol{x}_i) f(\boldsymbol{x}_i; H)\ . \tag{9}$$

and $f(\boldsymbol{x}; H)$ is the decision function of H defined in (1).

Proof. Based on the definitions in (3) and (4), it is not hard to obtain

$$\begin{aligned}
\mathtt{div}(H') &= 1 - \frac{1}{n(n+1)}\left(\sum\nolimits_{1\le i\ne j\le n} \mathtt{diff}(h_i, h_j) + 2\sum\nolimits_{i=1}^{n} \mathtt{diff}(h', h_i)\right) \\
&= 1 - \frac{1}{n(n+1)}\left(n(n-1)(1-\mathtt{div}(H)) + \frac{2}{m}\sum\nolimits_{i=1}^{m}\left(h'(\boldsymbol{x}_i)\sum\nolimits_{k=1}^{n} h_k(\boldsymbol{x}_i)\right)\right) \\
&= \frac{2}{n+1} + \frac{n-1}{n+1}\,\mathtt{div}(H) - \frac{2}{m(n+1)}\sum\nolimits_{i=1}^{m} h'(\boldsymbol{x}_i) f(\boldsymbol{x}; H)\ ,
\end{aligned}$$

which leads to the result directly. □

Algorithm 1. The DREP method

Input: ensemble to be pruned $H = \{h_i(\boldsymbol{x})\}_{i=1}^{n}$, validation data set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ and tradeoff parameter $\rho \in (0, 1)$

Output: pruned ensemble H^*

1: initialize $H^* \leftarrow \emptyset$
2: $h(\boldsymbol{x}) \leftarrow$ the classifier in H with the lowest error on S
3: $H^* \leftarrow \{h(\boldsymbol{x})\}$ and $H \leftarrow H \setminus \{h(\boldsymbol{x})\}$
4: **repeat**
5: **for** each $h'(\boldsymbol{x})$ in H **do**
6: compute $d_{h'} \leftarrow \mathtt{diff}(h', H^*)$ based on (9)
7: **end for**
8: sort classifiers $h'(\boldsymbol{x})'s$ in H in the ascending order of $d_{h'}$'s
9: $\Gamma \leftarrow$ the first $\lceil \rho \cdot |H| \rceil$ classifiers in the sorted list
10: $h(\boldsymbol{x}) \leftarrow$ the classifier in Γ which most reduces the error of H^* on S
11: $H^* \leftarrow \{h(\boldsymbol{x})\}$ and $H \leftarrow H \setminus \{h(\boldsymbol{x})\}$
12: **until** the error of H^* on S cannot be reduced

It can be found that at each step of greedy forward pruning, $\mathtt{div}(H)$ is a constant and $\mathtt{div}(H')$ is a monotonically decreasing function of $\mathtt{diff}(h', H)$. Thus, at each step, the task of estimating diversity $\mathtt{div}(H')$ can be substituted by computing the difference $\mathtt{diff}(h', H)$. The candidate classifier h' that can achieve smaller $\mathtt{diff}(h', H)$ will lead to larger diversity $\mathtt{div}(H')$. Obviously, in such a manner, we only need to compute the difference between the candidate classifier and the decision function of existing sub-ensemble rather than each of its members, which reduces the computational cost heavily comparing with the computation of diversity from scratch.

The pseudocode of the DREP method is presented in Algorithm 1. Specifically, it has three inputs: the ensemble H to be pruned, the validate data set S which is used to estimate empirical error and diversity and the tradeoff parameter ρ. Starting with the classifier with lowest error on validation set (lines 2-3), the DREP method iteratively selects classifier based on both empirical error and diversity. Concretely, at each step it first sorts the candidate classifiers in the ascending order of their differences with current sub-ensemble (lines 5-8), and then from the front part of sorted list it selects the classifier which can most reduce the empirical error on the validate data set. It can be found from Proposition 1 that the front classifiers will lead to large ensemble diversity. Also, among the front classifiers the one which reduces the empirical error most will be selected, thus it can be expected that the obtained ensemble will have both large diversity and small empirical error. These two criteria are balanced by the parameter ρ, i.e., the fraction of classifiers that are considered when minimizing empirical error. Obviously, a large value of ρ means that more emphasis on the empirical error, while a small ρ pays more attention on the diversity.

5 Empirical Studies

In this section, we perform experiments to evaluate the proposed DREP method, also to validate the theoretical results.

5.1 Settings

In experiments, we use twenty binary classification data sets from the UCI repository [10], amongst which four data sets are generated from multi-class data sets, that is, *letter** classifies 'u' against 'v' on *letter*; *optdigits* classifies '01234' against '56789' on *optdigits*; *satimage** classifies labels '1' and '2' against those with '5' and '7' on *satimage*; and *vehicle** classifies 'bus' and 'opel' against 'van' and 'saab' on *vehicle*. Since these data sets are widely used benchmarks, we omit their summary information for clarity here.

The DREP method and several comparative methods are evaluated in a series of experiments. Specifically, each experiment is performed on one data set, and mainly involves the following steps:

1. Randomly split the data set into three parts: 1/3 as training set, 1/3 as validation set and the rest as test set;
2. Using Bagging [3] to build an ensemble of 100 CART decision trees [4] on the training set;
3. Prune the obtained ensemble by using ensemble pruning methods, whose parameters are determined on the validation set;
4. Evaluate the performance of pruned ensemble on the test set, also record the size of the pruned ensemble.

On each data set, each experiment is run for thirty times. At each time, the sizes of pruned ensemble and its error rates on test set are recorded, and finally the averaged results with standard deviation over multiple runs are reported.

In experiments, the comparative methods include two benchmark methods:

- Bagging [3]: it is the full ensemble of all the 100 CART trees;
- Best Individual (BI): it selects the individual classifier which has the best performance on the validation set.

Moreover, the following greedy forward ensemble pruning methods are implemented and compared:

- Reduce-Error (RE) [19,6]: it starts with the classifier with lowest error, and then greedily selects the classifier that reduces error most;
- Kappa [19,1]: it starts with the pair of classifiers with lowest κ-statistics, and then iteratively adds the classifier with lowest κ-statistics with respect to current sub-ensemble;
- Complementarity (CP) [21,20]: this method starts with the classifier with lowest error, it incorporates at each iteration the one which is most complementary to the sub-ensemble;

Table 1. Error rates (mean±std.) achieved by comparative methods. On each data set an entry is marked with bullet '•' (or circle '○') if it is significantly better (or worse) than unpruned Bagging based on paired t-test at the significance level 0.1; the win/tie/loss counts are summarized in the last row.

Data set	Bagging	BI	RE	Kappa	CP	MD	DREP
australian	.134±.019	.148±.023○	.133±.015	.138±.017	.130±.014	.133±.016	.129±.016
breast-cancer	.278±.043	.293±.051	.275±.035	.280±.033	.288±.031	.311±.052○	.265±.024•
breast-w	.040±.010	.050±.012○	.034±.009•	.041±.011	.036±.009•	.035±.008	.034±.008•
diabetes	.239±.023	.256±.023○	.236±.022	.249±.020○	.240±.020	.243±.020	.234±.017
germen	.247±.016	.292±.021○	.248±.021	.254±.022	.250±.017	.247±.019	.248±.015
haberman	.261±.026	.270±.030○	.257±.025	.258±.034	.267±.029	.283±.037○	.252±.021
heart-statlog	.204±.039	.226±.041	.194±.034	.203±.035	.188±.034•	.195±.033	.183±.027•
hepatitis	.165±.030	.206±.049○	.164±.037	.183±.029○	.162±.031	.170±.036	.159±.027
ionosphere	.088±.024	.106±.034○	.069±.018•	.089±.030	.070±.019•	.079±.024	.066±.017•
kr-vs-kp	.014±.005	.013±.005	.009±.002•	.017±.005	.012±.004•	.015±.004	.008±.002•
*letter**	.047±.009	.075±.013○	.039±.008•	.047±.008	.039±.008•	.044±.008	.035±.007•
liver-dis	.313±.030	.362±.041○	.312±.032	.327±.039	.313±.035	.323±.038	.311±.029
*optdigits**	.046±.005	.109±.008○	.041±.004•	.045±.005	.040±.004•	.044±.005	.040±.003•
*satimage**	.032±.004	.051±.007○	.031±.004	.033±.005	.030±.004•	.032±.004	.029±.004•
sick	.016±.003	.017±.004	.015±.003	.017±.003	.015±.003	.016±.003	.014±.002•
sonar	.245±.050	.285±.036○	.235±.044	.245±.051	.216±.038•	.233±.044	.230±.030•
spambase	.071±.005	.094±.007○	.066±.005•	.070±.004	.066±.004•	.069±.005•	.066±.004•
tic-tac-toe	.060±.018	.101±.021○	.039±.008•	.082±.026○	.043±.010•	.078±.022○	.038±.007•
*vehicle**	.207±.020	.235±.029○	.207±.021	.214±.023	.204±.019	.215±.025	.203±.019
vote	.038±.011	.043±.014	.035±.011	.041±.016	.035±.013	.037±.011	.033±.007•
win/tie/loss	–	0/5/15	7/13/0	0/17/3	10/10/0	1/16/3	13/7/0

– Margin Distance (MD) [21,20]: at each iteration, this method incorporates into the ensemble tho classifier that reduces the distance from the margin vector to the objective point in the first quadrant the most.

It is easy to find that RE and Kappa consider only empirical error and diversity respectively, while CP, MD and DREP take both of them into account. For each ensemble pruning method, we will stop and return the sub-ensemble if its error rate on validation set cannot be reduced. In the experiments, the κ-statistics, complementarity measure, margin distance are estimated on the validation set, and the parameter ρ of DREP is selected in $\{0.2, 0.25, \ldots, 0.5\}$ on validation set.

All the experiments are run on a PC with 2GB memory.

5.2 Results

The error rates achieved by comparative methods are shown in Table 1. On each data set, paired t-test at significance level 0.1 is performed to compare performance of BI and ensemble pruning methods with that of Bagging. In Table 1, an entry is marked with bullet '•' (or circle '○') if it is significantly better (or worse) than Bagging, and the win/tie/loss counts are summarized in the last row. From the results, it is shown that BI which selects the best performed individual loses at 15 out of 20 data sets against Bagging, this coincides with the fact that ensemble usually achieves better performance than a single classifier. Meanwhile, the performance of ensemble pruning methods is much better. Specifically, RE,

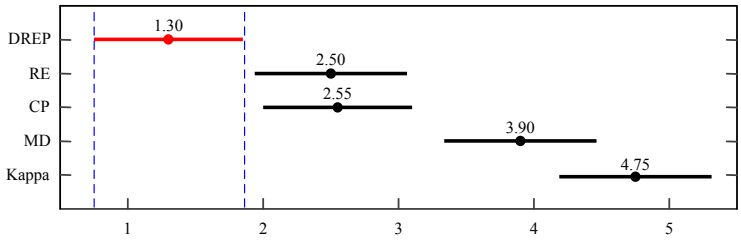


Fig. 1. The result of the Freidman test for comparing the performance of five ensemble pruning methods on 20 data sets. The dots indicate the average ranks, the bars indicate the critical difference with the Bonferroni-Dunn test at significance level 0.1, and compared methods having non-overlapped bars are significantly different.

CP and DREP respectively achieve 7, 10 and 13 wins but no losses compared with Bagging, while Kappa and MD respectively make 17 and 16 ties and only 3 losses. At the same time, from Table 2 it can be seen that the ensemble sizes are reduced from 100 to about 20. Hence, the purpose of ensemble pruning (that is, reduce ensemble size whilst keeping or improving performance) is reached; also amongst comparative ensemble pruning methods, it appears that DREP method performs quite well (it achieves the best win/tie/loss counts and the smallest average ensemble size).

To better compare the performance of greedy ensemble pruning methods, we perform *Freidman test* [8], which is a non-parametric statistical significance test for comparing multiple classifiers on multiple data sets. Roughly speaking, the Freidman test is based on the ranks of compared methods on multiple data sets, and it is performed in conjunction with the *Bonferroni-Dunn test* at certain significance level. Here, we employ it to compare the five greedy ensemble pruning methods used in our experiments, and the result is shown in Fig. 1. Amongst the five pruning methods, the DREP method gets the highest average rank (1.30), followed by RE (2.50) and CP (2.55), while the average ranks of MD and CP are 3.90 and 4.75 respectively. Since the critical difference with the two-tailed Bonferroni-Dunn test for 5 classifiers on 20 data sets is $2.241\sqrt{(5 \cdot 6)/(6 \cdot 20)} \approx 1.121$, we can find that the performance of DREP is significantly better than other methods (in Fig. 1 the bar of DREP is not overlapping with either of other methods). It is easy to understand that the performance of DREP is better than that of RE and Kappa, because RE and Kappa only consider the empirical risk and the diversity respectively while DREP take both of them into account. Also, RE performs significantly better than Kappa, which may imply that empirical error play a more important role in the tradeoff. This coincides with our theoretical results, because diversity plays a role of regularization which is used to prevent overfitting, and only considering regularization usually does not help to improve the generalization performance. Furthermore, it can be seen that DREP performs better than CP and MD, this can be explained that DREP explicitly tradeoffs empirical error and diversity regularization, while CP and MD implicitly consider the tradeoff at a fixed level and can be easily affected by noises.

Table 2. Ensemble sizes (mean±std.) of the pruned ensemble. On each data set, the entry achieving the smallest ensemble size is bolded, and the averaged sizes over all data sets are given in the last row.

Data set	RE	Kappa	CP	MD	DREP
australian	**15.4±3.5**	18.7±5.5	18.0±4.1	19.9±6.4	18.3±4.2
breast-cancer	18.4±3.8	22.0±9.2	18.1±5.1	24.1±10.0	**18.1±4.5**
breast-w	17.5±4.3	**15.5±3.9**	20.1±6.3	23.1±8.5	17.1±4.3
diabetes	23.5±5.8	26.7±11.4	21.9±6.4	29.0±10.1	**17.6±4.7**
germen	21.5±5.9	25.9±8.6	20.5±5.9	28.5±10.3	**17.1±3.9**
haberman	16.7±4.8	**15.7±5.0**	21.3±6.7	22.8±8.5	18.0±4.4
heart-statlog	18.9±4.1	22.9±8.1	21.9±5.8	23.5±7.8	**17.2±4.4**
hepatitis	14.4±2.6	**12.7±3.3**	17.7±5.3	18.9±5.9	17.7±4.6
ionosphere	**15.6±2.6**	21.8±9.0	19.5±5.8	23.8±7.7	17.5±4.6
kr-vs-kp	**16.1±3.7**	22.2±10.0	23.1±5.8	21.5±6.6	17.7±4.1
*letter**	22.9±4.6	25.0±7.6	**22.0±5.2**	27.8±9.2	24.7±4.5
liver-dis	22.7±5.7	23.6±10.9	21.5±5.6	25.5±9.3	**18.5±4.6**
*optdigits**	31.9±6.6	37.4±10.8	28.1±5.0	37.7±10.7	**25.1±4.8**
*satimage**	25.1±5.5	32.5±9.6	23.3±5.8	30.9±9.0	**24.8±4.7**
sick	**15.8±3.2**	22.7±10.9	20.9±4.7	22.8±9.0	17.7±4.4
sonar	20.1±5.0	22.1±9.8	21.1±5.9	24.9±9.5	**18.7±4.9**
spambase	25.3±6.8	27.5±8.6	24.3±7.0	28.4±8.3	**18.1±4.7**
tic-tac-toe	24.2±4.4	36.4±17.1	24.1±5.9	38.3±16.4	**18.9±4.3**
*vehicle**	**22.0±5.3**	25.7±8.5	22.3±7.1	25.2±9.4	24.9±4.9
vote	**12.8±1.6**	15.9±5.7	16.6±5.3	18.8±6.0	18.3±4.3
average	20.0	23.6	21.3	25.8	**19.3**

Table 2 presents the sizes of pruned ensembles, which shows that all the greedy pruning methods heavily reduce the ensemble sizes. Moreover, it can be seen that DREP achieves the smallest sizes on 10 data sets, also the smallest average ensemble size.

Furthermore, Fig. 2 plots the test error curves of Bagging and compared ensemble pruning methods on *heart-statlog* and *letter**. In detail, for Bagging the individual classifiers are aggregated in random order, and for ensemble pruning methods the greedy selection process will not be stopped until all the individuals are included, that is, the individual classifiers are aggregated in an order specified by the pruning methods. At each ensemble size the error is estimated on the test data, and the final results are obtained by averaging results of thirty runs, and they are plotted against ensemble sizes in Fig. 2. It can be seen that as ensemble size increases, the test error of Bagging decreases and converges, but the test errors of greedy ensemble pruning methods decrease much faster and are lower than Bagging, which indicates that better performance can be achieved at smaller ensemble sizes by using greedy ensemble pruning methods. By comparing the curves of DREP and other pruning methods, we can find that the test error of DREP decrease faster than other methods, even faster than RE which selects individual classifiers based on empirical error on the validation data set. This is not hard to understand because RE may overfit the validation data, while the diversity regularization used by DREP tends to help it achieve better performance.

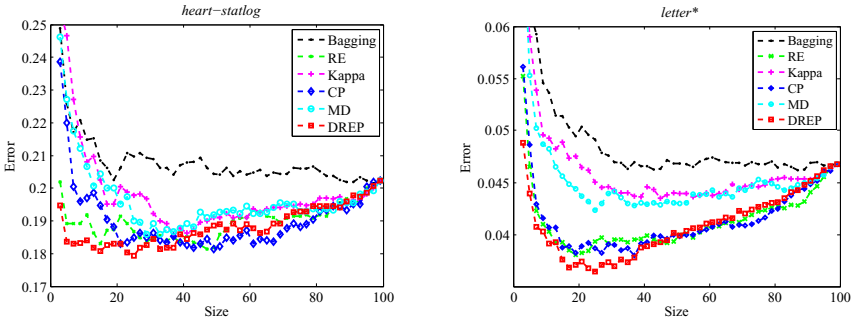


Fig. 2. Averaged test errors curves of Bagging and compared ensemble pruning methods on (a) *heart-statlog* and (b) *letter**, where horizontal axis and vertical axis correspond to ensemble size and test error respectively. For ensemble pruning methods, we do not stop the greedy selection process until all the individual classifiers are included.

In summary, we can see that with the help of diversity regularization, DREP is able to achieve significantly better generalization performance with smaller ensemble size than the compared methods.

6 Conclusion and Future Work

In ensemble learning, understanding diversity is one of the most important fundamental issues. This work focuses on the most popular setting of ensemble pruning, where the individual classifiers are combined by voting and the diversity is measured in the pairwise manner. In the PAC-learning framework, it presents a theoretical analysis on the role of diversity in voting, which is, to our best knowledge, the first PAC-style analysis on the effect of diversity in voting. It discloses that by enforcing large diversity, the hypothesis space complexity of voting can be reduced, and then better generalization performance can be expected. In the view of statistical learning, this implies that encouraging diversity can be regarded to apply regularization on ensemble methods. This may introduce a novel perspective of diversity in ensemble learning. Guided by this result, a greedy ensemble pruning method called DREP is proposed to explicitly exploit diversity regularization. Experimental results show that with the help of diversity regularization, DREP is able to achieve significantly better generalization performance with smaller ensemble size than the compared methods.

The current work applies diversity regularization on greedy ensemble pruning, it will be an interesting future work to develop ensemble learning methods which explicitly exploits diversity regularization. Recently it has been found that the ensemble diversity exists at multiple orders of correlation [5,34], thus it is also of great interest to study whether the theoretical results on diversity still hold in that case.

Acknowledgements. The authors want to thank anonymous reviewers for helpful comments. This research was supported by the NSFC (61021062, 60903103) and the 973 Program (2010CB327903).

References

1. Banfield, R., Hall, L., Bowyer, K., Kegelmeyer, W.: Ensemble diversity measures and their application to thinning. Information Fusion 6(1), 49–62 (2005)
2. Bartlett, P.: The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. IEEE Transactions on Neural Networks 44(2), 525–536 (1998)
3. Breiman, L.: Bagging predictors. Machine Learning 24(3), 123–140 (1996)
4. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey (1984)
5. Brown, G.: An Information Theoretic Perspective on Multiple Classifier Systems. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 344–353. Springer, Heidelberg (2009)
6. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: Proceedings of the 21st International Conference on Machine Learning, pp. 18–25 (2004)
7. Chen, H., Tiňo, P., Yao, X.: Predictive ensemble pruning by expectation propagation. IEEE Transactions on Knowledge and Data Engineering 21(7), 999–1013 (2009)
8. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
9. Dietterich, T.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning 40(2), 139–157 (2000)
10. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
11. Fumera, G., Roli, F.: A theoretical and experimental analysis of linear combiners for multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(6), 942–956 (2005)
12. Giacinto, G., Roli, F., Fumera, G.: Design of effective multiple classifier systems by clustering of classifiers. In: Proceedings of the 15th International Conference on Pattern Recognition, Barcelona, Spain, pp. 160–163 (2000)
13. Hernández-Lobato, D., Martínez-Muñoz, G., Suárez, A.: Empirical analysis and evaluation of approximate techniques for pruning regression bagging ensembles. Neurocomputing 74(12-13), 2250–2264 (2011)
14. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Advances in Neural Information Processing Systems, Denver, CO, vol. 7, pp. 231–238 (1994)
15. Kuncheva, L., Whitaker, C.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning 51(2), 181–207 (2003)
16. Kuncheva, L., Whitaker, C., Shipp, C., Duin, R.: Limits on the majority vote accuracy in classifier fusion. Pattern Analysis & Applications 6(1), 22–31 (2003)
17. Lazarevic, A., Obradovic, Z.: Effective pruning of neural network classifier ensembles. In: Proceedings of the IEEE/INNS International Joint Conference on Neural Networks, Washington, DC, pp. 796–801 (2001)

18. Li, N., Zhou, Z.-H.: Selective Ensemble under Regularization Framework. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 293–303. Springer, Heidelberg (2009)
19. Margineantu, D., Dietterich, T.: Pruning adaptive boosting. In: Proceedings of the 14th International Conference on Machine Learning, Nashville, TN, pp. 211–218 (1997)
20. Martínez-Muñoz, G., Hernández-Lobato, D., Suárez, A.: An analysis of ensemble pruning techniques based on ordered aggregation. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2), 245–259 (2009)
21. Martínez-Muñoz, G., Suárez, A.: Aggregation ordering in bagging. In: Proceeding of the IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria, pp. 258–263 (2004)
22. Martínez-Muñoz, G., Suárez, A.: Pruning in ordered bagging ensembles. In: Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, pp. 609–616 (2006)
23. Partalas, I., Tsoumakas, G., Vlahavas, I.: Focused ensemble selection: A diversity-based method for greedy ensemble selection. In: Proceedings of 18th European Conference on Artificial Intelligence, Patras, Greece, pp. 117–121 (2008)
24. Partalas, I., Tsoumakas, G., Vlahavas, I.: A study on greedy algorithms for ensemble pruning. Technical Report TR-LPIS-360-12, Department of Informatics, Aristotle University of Thessaloniki, Greece (2012)
25. Schapire, R., Freund, Y., Bartlett, P., Lee, W.: Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics 26(5), 1651–1686 (1998)
26. Tamon, C., Xiang, J.: On the Boosting Pruning Problem. In: Lopez de Mantaras, R., Plaza, E. (eds.) ECML 2000. LNCS (LNAI), vol. 1810, pp. 404–412. Springer, Heidelberg (2000)
27. Tang, E.K., Suganthan, P., Yao, X.: An analysis of diversity measures. Machine Learning 65(1), 247–271 (2006)
28. Tsoumakas, G., Partalas, I., Vlahavas, I.: An Ensemble Pruning Primer. In: Okun, O., Valentini, G. (eds.) Applications of Supervised and Unsupervised Ensemble Methods. SCI, vol. 245, pp. 1–13. Springer, Heidelberg (2009)
29. Valiant, L.: A theory of the learnable. Communications of the ACM 27, 1134–1142 (1984)
30. Yu, Y., Li, Y.-F., Zhou, Z.-H.: Diversity regularized machine. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Spain, pp. 1603–1608 (2011)
31. Zhang, T.: Covering number bounds of certain regularized linear function classes. Journal of Machine Learning Research 2, 527–550 (2002)
32. Zhang, Y., Burer, S., Street, W.: Ensemble pruning via semi-definite programming. Journal of Machine Learning Research 7, 1315–1338 (2006)
33. Zhou, Z.-H.: Ensemble Methods: Foundations and Algorithms. Chapman & Hall/CRC, Boca Raton, FL (2012)
34. Zhou, Z.-H., Li, N.: Multi-information Ensemble Diversity. In: El Gayar, N., Kittler, J., Roli, F. (eds.) MCS 2010. LNCS, vol. 5997, pp. 134–144. Springer, Heidelberg (2010)
35. Zhou, Z.-H., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. Artificial Intelligence 137(1-2), 239–263 (2002)

Ensembles on Random Patches

Gilles Louppe and Pierre Geurts

Dept. of EE & CS, & GIGA-R
University of Liège, Belgium

Abstract. In this paper, we consider supervised learning under the assumption that the available memory is small compared to the dataset size. This general framework is relevant in the context of big data, distributed databases and embedded systems. We investigate a very simple, yet effective, ensemble framework that builds each individual model of the ensemble from a random patch of data obtained by drawing random subsets of *both* instances and features from the whole dataset. We carry out an extensive and systematic evaluation of this method on 29 datasets, using decision tree-based estimators. With respect to popular ensemble methods, these experiments show that the proposed method provides on par performance in terms of accuracy while simultaneously lowering the memory needs, and attains significantly better performance when memory is severely constrained.

1 Motivation

Within the past few years, big data has become a popular trend among many scientific fields. In life sciences, computer vision, Internet search or finance, to cite a few, quantities of data have grown so large that it is increasingly difficult to process, analyze or visualize. In many cases, single computers are no longer fit for big data and distributed environments need to be considered to handle it. Although research is very active in this area, machine learning is no exception to this new paradigm. Much still needs to be done and methods and algorithms have to be reinvented to take this constraint into account.

In this context, we consider supervised learning problems for which the dataset is so large that it cannot be loaded into memory. In [1], Breiman proposed the Pasting method to tackle this problem by learning an ensemble of estimators individually built on random subsets of the training examples, hence alleviating the memory requirements since the base estimators would be built on only small parts of the whole dataset. Earlier, Ho proposed in [2] to learn an ensemble of estimators individually built on random subspaces (i.e., on random subsets of the features). While the first motivation of the Random Subspace method was to increase the diversity within the estimators of the ensemble, it can actually also be seen as way to reduce the memory requirements of building individual models. In this work, we propose to combine and leverage both approaches at the same time: learn an ensemble of estimators on *random patches*, i.e., on random subsets of the samples *and* of the features. Through an extensive empirical

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 346–361, 2012.

study, we show that this approach (1) improves or preserves comparable accuracy with respect to other ensemble approaches which build base estimators on the whole dataset while (2) drastically lowering the memory requirements and hence allowing an equivalent reduction of the global computing time.

The rest of this paper is organized as follows. Section 2 describes and then compares the Random Patches method with popular ensemble algorithms. In Section 3, we investigate experimentally the performance of the method on an extensive list of datasets and then draw some first conclusions. We then study in Section 4 the benefits of our algorithm under memory constraints and show that, in that context, it appears to be significantly better than other ensemble methods. We conclude and discuss future work directions in Section 5.

2 Random Patches

In this section, we formally describe our method, briefly introduce standard base estimators that have been considered in this work, and then discuss how our algorithm relates with popular ensemble methods.

2.1 Description

The Random Patches algorithm proposed in this work (further referred to as RP) is a wrapper ensemble method that can be described in the following terms. Let $R(p_s, p_f, D)$ be the set of all random patches of size $p_s N_s \times p_f N_f$ than can be drawn from the dataset D, where N_s (resp. N_f) is the number of samples in D (resp. the number of features in D) and where $p_s \in [0, 1]$ (resp. p_f) is an hyper-parameter that controls the number of samples in a patch (resp. the number of features). That is, $R(p_s, p_f, D)$ is the set of all possible subsets containing $p_s N_s$ samples (among N_s) with $p_f N_f$ features (among N_f). The method then works as follows:

1. Draw a patch $r \sim U(R(p_s, p_f, D))$ uniformly at random.
2. Build an estimator on the selected patch r.
3. Repeat 1-2 for a preassigned number T of estimators.
4. Aggregate the predictions by voting (in case of classifiers) or averaging (in case of regressors) the predictions of the T estimators.

2.2 Tree-Based Methods

While the RP algorithm can exploit any kind of base estimators, we consider in this work only tree-based estimators. We first describe standard classification and regression trees and ensemble methods and then the two specific base learners we have considered in our experiments.

Classification and Regression Trees. A standard classification/regression tree [3] is an input-output model represented by a tree. Internal nodes of the tree are labeled with a (usually binary) test based on one input feature. Leaves are labeled with a value of the output (discrete or continuous). The predicted

output for a new instance is determined as the output associated to the leaf reached by the instance when it is propagated through the tree. A decision tree is built using a recursive procedure which identifies at each node the test that leads to a split of the node sample into two subsamples that are as pure as possible in terms of their output values, as measured by a so-called score measure. The construction of the tree then stops when some stopping criterion is met.

Ensemble of Randomized Trees. Single decision trees typically suffer from high variance, which makes them not competitive in terms of accuracy. A very efficient and simple way to address this flaw is to use them in the context of randomization-based ensemble methods. Specifically, the core principle is to introduce random perturbations into the learning procedure in order to produce several different decision trees from a single learning set. For example, in Bagging [4], trees are built on randomly drawn bootstrap copies of the original data, hence producing different decision trees. In Random Forests [5] (RF), Bagging is extended and combined with a randomization of the input features that are used when considering candidates to split internal nodes. In particular, instead of looking for the best split among all features, the algorithm selects, at each node, a random subset of K features and then determines the best test over these features only. In Extremely Randomized Trees [6] (ET), randomization goes even one step further: discretization threholds are also drawn at random and the best test is chosen among the K randomly drawn cut-points. Unlike in RF though, the trees in ET are not built on bootstrap copies of the input data.

Base Estimators. We consider and evaluate two base estimators within the RP algorithm: standard classification trees and (single) extremely randomized trees. Unless otherwise stated, trees are unpruned and grown using Gini entropy as the main scoring criterion for node splitting. The parameter K of extremely randomized trees within RP is set to its maximum value $K = p_f N_f$ (i.e., corresponding to no further random selection of features).

2.3 Related Work

The first benefit of RP is that it generalizes both the Pasting Rvotes (P) method [1] (and its extensions [7,8]) and the Random Subspace (RS) algorithm [2]. Both are indeed merely particular cases of RP: setting $p_s = 1.0$ yields RS while setting $p_f = 1.0$ yields P. As such, it is expected that when both hyper-parameters p_s and p_f are tuned, RP should be at least as good as the best of the two methods, provided there is no overfitting associated with this tuning.

When the base estimators are standard decision trees (resp. extremely randomized trees with $K = p_f N_f$), interesting parallels can also be drawn between RP and the RF algorithm (resp. ET). For $p_s = 1.0$, the value of $p_f N_f$ is indeed nearly equivalent to the number K of features randomly considered when splitting a node. A major difference remains though. In RP, the subset of features is selected globally once and for all, prior to the construction of the tree. By

contrast, in RF (resp. in ET) subsets of features are drawn locally at each node. Clearly, the former approach already appears to be more attractive when dealing with large databases. Non-selected features indeed do not need to be considered at all, hence lowering the memory requirements for building a single tree. Another interesting parallel can be made when bootstrap samples are used like in RF: it nearly amounts to set $p_s = 0.632$, i.e. the average proportion of unique samples in a bootstrap sample. Differences are that in a bootstrap sample, the number of unique training samples varies from one to another (while it would be fixed to $0.632N_s$ in RP), and that samples are not all equally weighted.

In addition, RP also closely relates to the SubBag algorithm [9] which combines Bagging and RS for constructing ensembles. Using N_s bootstrapped samples (i.e., nearly equivalent to $p_s = 0.632$) and setting $p_f = 0.75$, Panov et al showed that SubBag has comparable performance to that of RF. An added advantage of SubBag, and hence of RP, is that it is applicable to any base estimator without the need to randomize the latter.

3 On Accuracy

Our validation of the RP algorithm is carried out in two steps. In this section, we first investigate how RP compares with other popular tree-based ensemble methods in terms of accuracy. In the next section, we then focus on its memory requirements for achieving optimal accuracy and its capability to handle strong memory constraints, again in comparison with other ensemble methods.

Considering accuracy only, our main objective is to investigate whether the additional degrees of freedom brought by p_s and p_f significantly improve, or degrade, the performance of RP. Additionally, our goal is also to see whether sampling features once globally, instead of locally at each node, impairs performance, as this is the main difference between RP and state-of-the-art methods such as RF or ET.

3.1 Protocol

We compare our method with P and RS, as well as with RF and ET. For RP, P and RS, two variants have been considered, one using standard decision trees (suffixed below with '-DT') as base estimators, and the other using extremely randomized trees (suffixed below with '-ET') as base estimators. Overall, 8 methods are compared: RP-DT, RP-ET, P-DT, P-ET, RS-DT, RS-ET, RF and ET.

We evaluate the accuracy of the methods on an extensive list of both artificial and real classification problems. For each dataset, three random partitions were drawn: the first and larger (50% of the original dataset) to be used as the training set, the second (25%) as validation set and the third (25%) as test set. For all methods, the hyper-parameters p_s and p_f were tuned on the validation set with a grid-search procedure, using the grid $\{0.01, 0.1, ..., 0.9, 1.0\}$ for both p_s and p_f. All other hyper-parameters were set to default values. In RF and ET, the number K of features randomly selected at each node was tuned using the grid $p_f N_f$. For all ensembles, 250 fully developed trees were generated

and the generalization accuracy was estimated on the test set. Unless otherwise mentioned, for all methods and for all datasets, that procedure was repeated 50 times, using the same 50 random partitions between all methods, and all scores reported below are averages over those 50 runs. All algorithms and experiments have been implemented in Python, using Scikit-Learn [10] as base framework.

3.2 Small Datasets

Before diving into heavily computational experiments, we first wanted to validate our approach on small to medium datasets. To that end, experiments were carried out on a sample[1] of 16 well-known and publicly available datasets (see Table 1) from the UCI machine learning repository [11]. Overall, these datasets cover a wide range of conditions, with the sample sizes ranging from 208 to 20000 and the number of features varying from 6 to 168. Detailed average performances of the 8 methods for all 16 datasets using the protocol described above are reported in Table 1 of the supplementary materials[2]. Below, we analyze general trends by performing various statistical tests.

Following recommendations in [12], we first performed a Friedman test that rejected the hypothesis that all algorithms are equivalent at a significance level $\alpha = 0.05$. We then proceeded with a post-hoc Nemenyi test for a pairwise comparison of the average ranks of all 8 methods. According to this test, the performance of two classifiers is significantly different (at $\alpha = 0.05$) if their average ranks differ by at least the critical difference $CD = 2.6249$ (See [12] for further details). The diagram of Figure 1 summarizes these comparisons. The top line in the diagram is the axis along which the average rank R_m of each method m is plotted, from the highest ranks (worst methods) on the left to the lowest ranks (best methods) on the right. Groups of methods that are not statistically different from each other are connected. The critical difference CD is shown above the graph. To further support these rank comparisons, we also compare the 50 accuracy values obtained over each dataset split for each pair of methods by using a paired t-test (with $\alpha = 0.01$). The results of these comparisons are summarized in Table 2 in terms of "Win-Draw-Loss" statuses of all pairs of methods; the three values at the intersection of row i and column j of this table respectively indicate on how many datasets method i is significantly better/not significantly different/significantly worse than method j.

Since all methods are variants of ensembles of decision trees, average accuracies are not strikingly different from one method to another (see Table 1 of the supplementary materials). Yet, significant trends appear when looking at Figure 1 and Table 2. First, all ET-based methods are ranked before DT-based methods, including the popular Random Forest algorithm. Overall, the original ET algorithm is ranked first ($R_{ET} = 2.125$), then come RS-ET and RP-ET at close positions ($R_{RS-ET} = 2.8125$ and $R_{RP-ET} = 2.9375$) while P-ET is a bit behind ($R_{P-ET} = 3.75$). According to Figure 1, only ET is ranked significantly higher than all DT-based method but looking at Table 2, the worse ET-based variant

[1] These datasets were chosen a priori and independently of the results obtained.

[2] http://www.montefiore.ulg.ac.be/~glouppe/pdf/ecml12-suppl.pdf

Table 1. Small datasets

Dataset	N_s	N_f
DIABETES	768	8
DIG44	18000	16
IONOSPHERE	351	34
PENDIGITS	10992	16
LETTER	20000	16
LIVER	345	6
MUSK2	6598	168
RING-NORM	10000	20
SATELLITE	6435	36
SEGMENT	2310	19
SONAR	208	60
SPAMBASE	4601	57
TWO-NORM	9999	20
VEHICLE	1692	18
VOWEL	990	10
WAVEFORM	5000	21

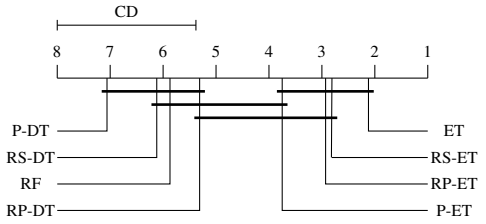


Fig. 1. Average ranks of all methods

Table 2. Pairwise t-test comparisons

	RF	ET	P-DT	P-ET	RS-DT	RS-ET	RP-DT	RP-ET
RF	—	1/2/13	12/4/0	1/7/8	4/7/5	2/2/12	1/10/5	0/4/12
ET	13/2/1	—	14/1/1	10/5/1	13/3/0	4/11/1	12/2/2	5/10/1
P-DT	0/4/12	1/1/14	—	0/4/12	2/3/11	2/1/13	0/4/12	0/4/12
P-ET	8/7/1	1/5/10	12/4/0	—	9/6/1	2/6/8	9/6/1	0/11/5
RS-DT	5/7/4	0/3/13	11/3/2	1/6/9	—	0/2/14	1/11/4	0/4/12
RS-ET	12/2/2	1/11/4	13/1/2	8/6/2	14/2/0	—	11/4/1	1/13/2
RP-DT	5/10/1	2/2/12	12/4/0	1/6/9	4/11/1	1/4/11	—	0/6/10
RP-ET	12/4/0	1/10/5	12/4/0	5/11/0	12/4/0	2/13/1	10/6/0	—

(P-ET) is still 9 times significantly better (w.r.t. the 50 runs over each set) and only 1 times significantly worse than the best DT-based variant (RP-DT). The separation between these two families of algorithm thus appears quite significant. This observation clearly suggests that using random split thresholds, instead of optimized ones like in decision trees, pays off in terms of generalization.

Among ET-based methods, RP-ET is better than P-ET but it is superseded by ET and RS-ET in terms of average rank. Since RS-ET is a particular case of RP-ET, this suggests that we are slightly overfitting when tuning the additional parameter p_s. And indeed RP-ET is better ranked than RS-ET in average on the validation set (results not shown). Table 2 however indicates otherwise and makes RP-ET appear as slightly better than RS-ET (2/13/1). Regarding ET over RP-ET, the better performance of the former (5/10/1) is probably due to the fact that in ET subsets of features are redrawn locally at each node when building trees and not once and for all prior to their construction. This gives less chances to generate improper trees because of a bad initial choice of features and thus leads to a lower bias and a better accuracy.

Among DT-based methods, RP-DT now comes first (mean rank of 5.3125), then RF ($R_{RF} = 5.875$), RS-DT ($R_{RS-DT} = 6.125$) and then P-DT in last ($R_{P-DT} = 7.0625$). RP is only significantly worse than another DT-based variant on 1 dataset. The extra-randomization brought by the random choices of both samples and features seems to be beneficial with decision trees that do not benefit from the randomization of discretization thresholds. The fact that RF samples features locally does not appear here anymore as an advantage over RP (RF is significantly worse on 5 problems and better on only one), probably

Table 3. Large datasets

Dataset	N_s	N_f
CIFAR10*	60000	3072
MNIST3VS8	13966	784
MNIST4VS9	13782	784
MNIST*	70000	784
ISOLET	7797	617
ARCENE	900	10000
BREAST2	295	24496
MADELON	4400	500
MARTI0	500	1024
REGED0	500	999
SECOM	1567	591
TIS	13375	927
SIDO0*	12678	4932

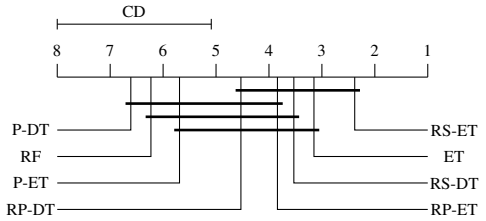


Fig. 2. Average ranks of all methods

Table 4. Pairwise t-test comparisons

	RF	ET	P-DT	P-ET	RS-DT	RS-ET	RP-DT	RP-ET
RF	—	1/5/7	8/3/2	2/6/5	0/6/7	0/5/8	0/6/7	0/6/7
ET	7/5/1	—	9/2/2	7/6/0	3/7/3	0/9/4	5/6/2	1/11/1
P-DT	2/3/8	2/2/9	—	1/5/7	0/3/10	0/3/10	1/3/9	0/4/9
P-ET	5/6/2	0/6/7	7/5/1	—	0/6/7	0/5/8	2/5/6	1/5/7
RS-DT	7/6/0	3/7/3	10/3/0	7/6/0	—	1/8/4	2/11/0	1/10/2
RS-ET	8/5/0	4/9/0	10/3/0	8/5/0	4/8/1	—	4/8/1	0/13/0
RP-DT	7/6/0	2/6/5	9/3/1	6/5/2	0/11/2	1/8/4	—	1/9/3
RP-ET	7/6/0	1/11/1	9/4/0	7/5/1	2/10/1	0/13/0	3/9/1	—

because the decrease of bias that it provides does not exceed the increase of variance with respect to global feature selection.

3.3 Larger Datasets

While the former experiments revealed promising results, it is fair to ask whether the conclusions that have been drawn would hold on and generalize to larger problems, for example when dealing with a few relevant features buried into hundreds or thousands of not important features (e.g., in genomic data), or when dealing with many correlated features (e.g., in images). To investigate this question, a second bench of experiments was carried out on 13 larger datasets (see Table 3). All but MADELON are real data. In terms of dimensions, these datasets are far bigger, ranging from a few hundreds of samples and thousands of features, to thousands of samples but hundreds of features. As such, the complexity of the problems is expected to be greater. We adopted the exact same protocol as for smaller datasets. However, to lower computing times, for datasets marked with $*$, the methods were run using 100 trees instead of 250 and the minimum number of samples required in an internal node was set to 10 in order to control complexity. Detailed results are provided in Table 2 of the supplementary materials and are summarized in Figure 2 and Table 4, respectively in terms of average rank (the critical difference at $\alpha = 0.05$ is now 2.9120) and Win/Draw/Loss statuses obtained with paired t-tests. A Friedman test (at $\alpha = 0.05$) still indicates that some methods are significantly different from the others.

As it may be observed from Figure 2, the average ranks of the methods are closer to each other than in the previous experiments, now ranging from 2.38 to 6.61, while they were previously ranging from 2.12 to 7. Methods are more

connected by critical difference bars. This suggests that overall they behave more similarly to each other than before. General trends are nevertheless comparable to what we observed earlier. ET-based methods still seem to be the front-runners. From Figure 2, RS-ET, ET and RP-ET are in the top 4, while P-DT, RF and RP-DT remain in the second half of the ranking. Surprisingly however, RS-DT now comes right after RS-ET and ET and just before RP-ET whereas it ranked penultimate on the smaller datasets. Table 4 however suggests that RS-DT performs actually a little worse against RP-ET (1/10/2). All in all, it thus still seems beneficial to randomize split thresholds on the larger datasets.

Comparing ET-based variants, ET is no longer the best method on average, but RS-ET is (with 4/9/0 for RS-ET versus ET). This suggests than on larger datasets, picking features globally at random prior to the construction of the trees is as good, or even beat picking them locally at each node. Due to the quantitatively larger number of samples in a patch, and also to the larger number of redundant features expected in some large datasets (e.g., in CIFAR10 or MNIST), it is indeed less likely to build improper trees with strong biases. As a result, variance can be further decreased by sampling globally. In support of this claim, on a few problems such as ARCENE, BREAST2, or MADELON that contain many irrelevant features, ET remains the best method. In that case, it is indeed more likely to sample globally improper random patches, and hence to build improper trees. The average rank of RP-ET suggests that it performs worse than RS-ET and thus that there is some potential overfitting when tuning p_s in addition to p_f. This difference is however not confirmed in Table 4 where the accuracies of these two methods are shown to be never significantly different (0/13/0). RP-ET is also on a perfect par with ET (1/11/1). Among DT-based variants, RP-DT, which was the best performer on small datasets, is still ranked above RF and P-DT, but it is now ranked below RS-DT with a win/draw/loss of 0/11/2. This is again due to some overfitting.

While less conclusive than before, the results on larger datasets are consistent with what we observed earlier. In particular, they indicate that the Random Patches method (with ET) remains competitive with the best performers.

3.4 Conclusions

Overall, this extensive experimental study reveals many interesting results. The first and foremost result is that ensembles of randomized trees nearly always beat ensembles of standard decision trees. As off-the-shelf methods, we advocate that ensembles of such trees should be preferred to ensembles of decision trees. In particular, these results show that the well-known Random Forest algorithm does not compete with the best performers. Far more important to our concern though, this study validates our RP approach. Building ensembles (of ET) on random patches of data is competitive in terms of accuracy. Overall, there is no strong statistical evidence that the method performs less well, but there is also no conclusive evidence that it significantly improves performance. Yet, results show that RP is often as good as the very best methods. Regarding the shape of the random patches, the strategy behind Pasting (i.e., p_s free and $p_f = 1.0$) proved

to be (very) ineffective on many datasets while the Random Subspace algorithm (i.e., $p_s = 1.0$ and p_f free) always ranked among the very best performers. On average, RS indeed came in second on the small datasets and in first on the larger datasets, which tends to indicate that sampling features is crucial in terms of accuracy. As for patches of freely adjustable size (i.e., using RP), they showed to be slightly sensitive to overfitting but proved to remain closely competitive with the very best methods. In addition, these results also suggest that sampling features globally, once and for all, prior to the construction of a (randomized) decision tree, does not actually impair performance. For instance, RS-ET or RP-ET are indeed not strongly worse, nor better, than ET, in which candidates features are re-sampled locally at each node.

4 On Memory

Section 3 reveals that building an ensemble of base estimators on random patches, instead of the whole data, is a competitive strategy. In the context of big data, that is when the size of the dataset is far bigger than the available memory, this suggests that using random parts of the data of the appropriate size to build each base estimator would likely result in an ensemble which is actually as good as if the whole data could have been loaded and used.

Formally, we assume a general framework where the number of data units that can be loaded at once into memory is constrained to be lower than a given threshold M_{max}. Not considering on-line algorithms within the scope of this study, M_{max} can hence be viewed as the total units of data allowed to be used to build a single base estimator. In the context of our sampling methods, the amount of memory required for a patch is given by $(p_s N_s)(p_f N_f)$ and thus constraining memory by M_{max} is equivalent to constraining the relative patch size $p_s p_f$ to be lower than $M'_{max} = M_{max}/(N_s N_f)$. While simplistic[3], this framework has the advantage of clearly addressing one of the main difficulties behind big data, that is the lack of fast memory. Yet, it is also relevant in other contexts, for example when data is costly to access (e.g., on remote locations) or when algorithms are run on embedded systems with strong memory constraints.

In Section 4.1, we first study the effects of p_s and p_f on the accuracy of the resulting ensemble and show that it is problem and base estimator dependent. Second, we show that the memory requirements, i.e., the relative size $p_s p_f$ of the random patches, can often be drastically reduced without significantly degrading the performance of the ensemble (Section 4.2). Third, because the sensitivity of the ensemble to p_s and p_f is problem and base estimator specific, we show that under very strong memory constraints adjusting both parameters at the same time, as RP does, is no longer merely as good but actually significantly better than other ensemble methods (Section 4.3).

[3] e.g., the quantity of memory used by the estimator itself is not taken into account.

4.1 Sensitivity to p_s and p_f

Let us first consider and analyze the sets $\{(p_s, p_f, Acc_D(p_s, p_f))|\forall p_s, p_f\}$ for various problems, where $Acc_D(p_s, p_f)$ is the average test accuracy of an ensemble built on random patches of size $p_s p_f$ (using the same protocol as previously) on the dataset D.

As Figure 3 illustrates for six datasets, the surfaces defined by these sets vary significantly from one problem to another. We observed four main trends. In Figures 3a, and 3b (resp. 3c), accuracy increases with p_s (resp. p_f) while adjusting p_f (resp. p_s) has no or limited impact. In Figure 3d, the best strategy is to increase both p_s and p_f. Finally, in Figures 3e and 3f, the surface features plateaus, which means that beyond some threshold, increasing p_s or p_f does not yield any significant improvement. Interestingly, in most of the cases, the optimum corresponds to a value $p_s p_f$ much smaller than 1.

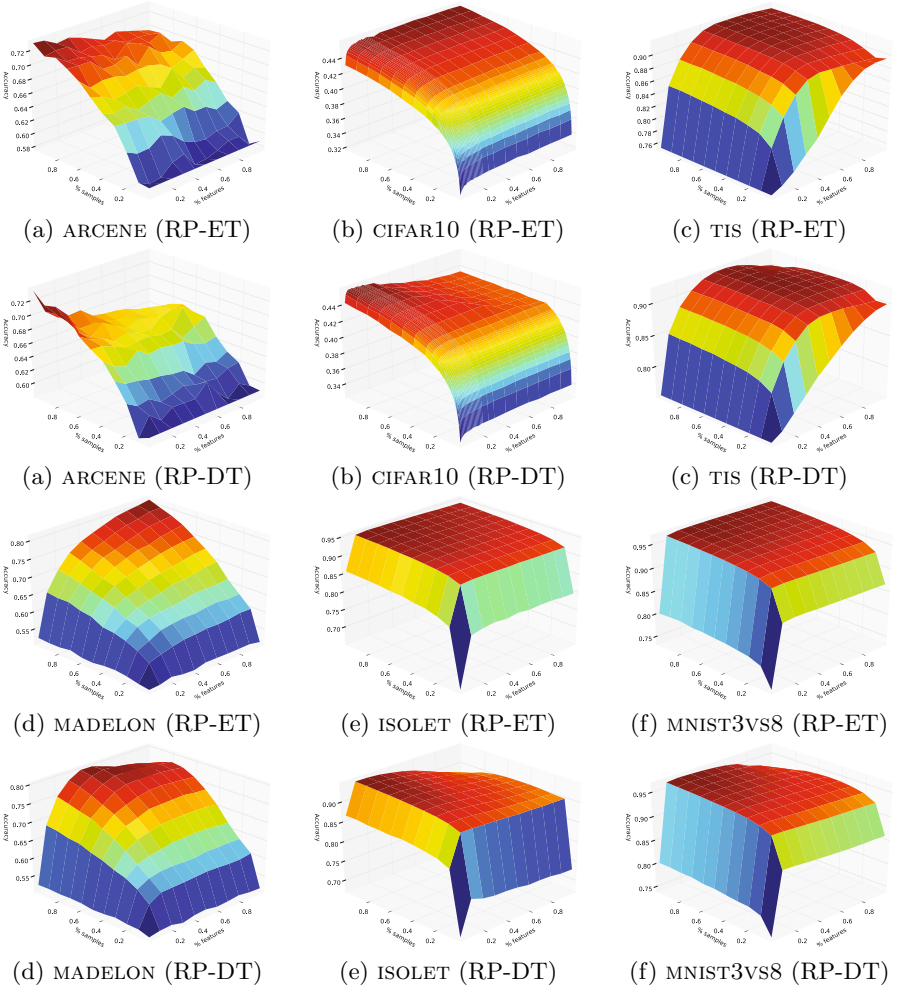
The choice of the base estimators does not have a strong impact on the aspect of the curves (compare the 1^{st} and 3^{rd} rows of sub-figures in Figure 3 with those in the 2^{nd} and 4^{th} rows). The only difference is the decrease of the accuracy of RP-DT when p_s and p_f grow towards 1.0. Indeed, since the only source of randomization in RP-DT is patch selection, it yields in this case ensembles of identical trees and therefore amounts to build a single tree on the whole dataset. By contrast, because of the extra-randomization of the split thresholds in ET, there is typically no drop of accuracy for RP-ET when p_s and p_f grow to 1.0.

Overall, this analysis suggests that not only the best pair $p_s p_f$ depends on the problem, but also that the sensitivity of the ensemble to changes to the size of a random patch is both problem and base estimator specific. As a result, these observations advocate for a method that could favor p_s, p_f or both, and do so appropriately given the base estimator.

4.2 Memory Reduction, without Significant Loss

We proceed to study in this section the actual size of the random patches when the values of p_s and p_f are tuned using an independent validation set. Our results are summarized in Figure 4a. Each ellipse corresponds to one of the 29 datasets of our benchmark, whose center is located at $(\overline{p_s}, \overline{p_f})$ (i.e., the average parameter values over the 50 runs) and whose semi-axes correspond to the standard deviations of p_s and p_f. Any point in the plot corresponds to a pair (p_s, p_f) and thus to a relative consumption $M' = p_s p_f$ of memory. To ease readability, level curves are plotted for $M' = 0.01, 0.1, ..., 0.9$. In the right part of the figure, the histogram counts the number of datasets such that $\overline{p_s} \cdot \overline{p_f}$ falls in the corresponding level set.

Figure 4a corroborates our previous discussion. On some datasets, it is better to favor p_s while on some other increasing p_f is a better strategy. The various sizes of the ellipses also confirm that the sensitivity to variations of p_s and p_f is indeed problem-specific. The figure also clearly highlights the fact that, even under no memory constraint, the optimal patches rarely consume the whole memory. A majority of ellipses indeed lie below the level set $M' = 0.5$ and only a couple of them are above $M' = 0.75$. With respect to ET or RF for which the



(a) ARCENE (RP-ET) (b) CIFAR10 (RP-ET) (c) TIS (RP-ET)

(a) ARCENE (RP-DT) (b) CIFAR10 (RP-DT) (c) TIS (RP-DT)

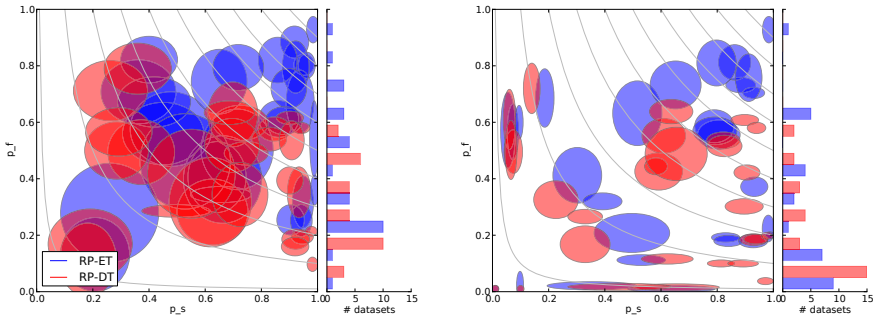
(d) MADELON (RP-ET) (e) ISOLET (RP-ET) (f) MNIST3VS8 (RP-ET)

(d) MADELON (RP-DT) (e) ISOLET (RP-DT) (f) MNIST3VS8 (RP-DT)

Fig. 3. Learning surfaces

base estimators are all built on the whole dataset, this means that ensembles of patches are not only as competitive but also less memory greedy. In addition, the figure also points out the difference between RP-ET and RP-DT as discussed in the previous section. To ensure diversity, RP-DT is constrained to use smaller patches than RP-ET, hence explaining why the ellipses in red are on average below those in blue. While RP-DT proved to be a bit less competitive in terms of accuracy, this indicates on the other hand that RP-DT may actually be more interesting from a memory consumption point of view.

In Section 4.1, we observed plateaus or very gentle slopes around the optimal pair (p_s, p_f). From a memory point of view, this suggests that the random



(a) Original patches (b) Reduced patches

Fig. 4. Optimal sizes of the random patches on our benchmark

patches are likely to be reducible without actually degrading the accuracy of the resulting ensemble. Put otherwise, our interest is to find the smallest size $p_s p_f$ such that the accuracy of the resulting ensemble is not significantly worse than an ensemble built without such constraint. To that end, we study the extent at which the constraint $p_s p_f < M'_{max}$ can be strengthened without any significant drop in accuracy. If M'_{max} can be reduced significantly then it would indeed mean that even when only small parts of the data are actually used to build single base estimators, competitive performance can still be achieved.

Figure 4b summarizes our results. For all datasets, M'_{max} was set to the lowest value such that it cannot be statistically detected that the average accuracy of the resulting ensemble is different from the average accuracy of an ensemble built with no memory constraint (at $\alpha = 0.05$). With regard to Figure 4a, the shift of most ellipses to lower memory level sets confirm our first intuition. In many cases, the size of the random patches can indeed be reduced, often drastically, without significant decrease of accuracy. For more than half of the datasets, memory can indeed be decreased to $M' = 0.1$ or $M' = 0.2$. In other words, building trees on small parts of the data (i.e., 10% or 20% of the original dataset) is, for more than half of the datasets, enough to reach competitive accuracy. Also, the sensitivity to p_s and p_f is now even more patent. Some ensembles use very few samples ($p_s < 0.1$) but with many features, while other uses many samples with few features ($p_f < 0.1$). Again, from a memory point of view, RP-DT appears to be more interesting than RP-ET. The memory reduction is larger, as the histogram indicates. Optimized splits in the decision trees may indeed lead to a better exploitation of the data, hence to a potentially larger reduction of memory. In conclusion, while not detecting significant differences in accuracy does not allow to conclude that the performances are truly similar, these figures at least illustrate that memory requirements can be drastically reduced without apparent loss in accuracy.

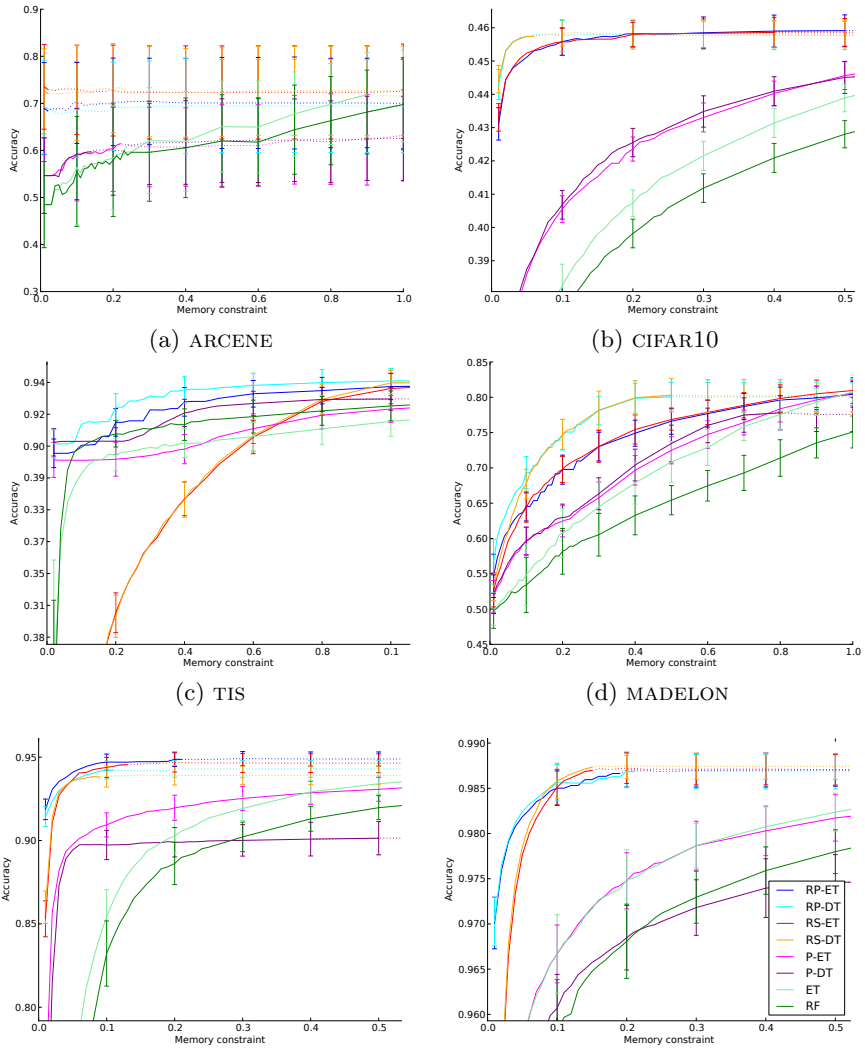
4.3 Memory Reduction, with Loss

The previous section has shown that the memory consumption can be reduced up to some threshold M'_{max} with no significant loss in accuracy. In this section we now look at the accuracy of the resulting ensemble when M'_{max} is further decreased. We argue that with severe constraints, and because datasets have all a different sensitivity, it is even more crucial to better exploit data and thus to find the right trade-off between both p_s and p_f, as only RP can.

To illustrate our point, Figure 5 compares for 6 representative datasets the accuracy of the methods with respect to the memory constraint $p_s p_f < M'_{max}$. A plain line indicates that the generalization error of the best resulting ensemble under memory constraint M'_{max} is significantly (at $\alpha = 0.05$) worse on the test sets than when there is no constraint (i.e., $M'_{max} = 1$). A dotted line indicates that on average, on the test set, the ensemble is not significantly less accurate.

As the figure shows, when M'_{max} is low, RP-based ensembles often achieve the best accuracy. Only on ARCENE (Figure 5a), RS seems to be a better strategy, suggesting some overfitting in setting p_s in RP. On all 5 other example datasets, RP is equivalent or better than RS and P for low values of M'_{max}, with the largest gaps appearing on ISOLET (Figure 5e) and MNIST3VS8 (Figure 5f). As already observed in the previous section, although RP-DT is not the best strategy when memory is unconstrained, its curve dominates the curve of RP-ET for small values of M'_{max} in Figures 5b, 5c, and 5d. Because split thresholds are not randomized in RP-DT, this method is more resistant than RP-ET to the strong randomization induced by a very low M'_{max} threshold.

For comparison, Figure 5 also features the learning curves of both ET and RF (with K optimized on the validation set), in which the trees have all been built on the *same* training sample of $M'_{max} N_s$ instances, with all features. These results are representative of the use of a straightforward sub-sampling of the instances to handle the memory constraint. On all datasets, this setting yields very poor performance when M'_{max} is low. Building base estimators on re-sampled random patches thus brings a clear advantage to RP, RS and P and hence confirms the conclusions of Basilico et al who showed in [8] that using more data indeed produces more accurate models than learning from a single subsample. This latter experiment furthermore shows that the good performances of RP cannot be trivially attributed to the fact that our datasets contain so many instances that only processing a subsample of them would be enough. On most problems, the slopes of the learning curves of RF and ET indeed suggest that convergence has not yet been reached on these datasets. Yet, important improvement are gained by sub-sampling random patches. Overall, these results thus indicate that building an ensemble on random patches is not only a good strategy when data is abundant and redundant but also that it works even for scarce datasets with limited information regarding the problem.



(a) ARCENE (b) CIFAR10

(c) TIS (d) MADELON

(e) ISOLET (f) MNIST3VS8

Fig. 5. Accuracy under memory constraint

4.4 Conclusion

We have shown in this section that the memory requirements of sampling-based ensembles are intrinsically low. Better, we have shown that they can often be drastically decreased without significant loss in accuracy. When the size of the dataset is far bigger than the available memory, we have also demonstrated that sampling data along both samples and features, as RP does, not only competes with other ensemble algorithms but also significantly improves the accuracy of the resulting ensemble. It also brings a significant improvement over a straightforward sub-sampling of the instances.

5 Conclusions and Future Work

The main contribution of this paper is to explore a new framework for supervised learning in the context of very strong memory constraints or, equivalently, very large datasets. To address such problems, we proposed the Random Patches ensemble method that builds each individual model of the ensemble from a random patch of the dataset obtained by drawing random subsets of both samples and features from the whole dataset. Through extensive experiments with tree-based estimators, we have shown that this strategy works as well as other popular randomization schemes in terms of accuracy (Section 3), at the same time reduces very significantly the memory requirements to build each individual model (Section 4.2), and, given its flexibility, attains significantly better accuracy than other methods when memory is severely constrained (Section 4.3). Since all models are built independently of each other, the approach is furthermore trivial to parallelize. All in all, we believe that the paradigm of our method highlights a very promising direction of research to address supervised learning on big data.

There remain several open questions and limitations to our approach that we would like to address in the future. First, this study motivates our interest in experimenting with truly large-scale problems (of giga-scale and higher). Since RP already appears advantageous for small to medium datasets, the potential benefits on very large-scale data indeed look very promising.

Second, the conclusions drawn in sections 3 and 4 are all based on the optimal values of the parameters p_s and p_f tuned through an exhaustive grid search on the validation set. Our analyses did not account for the memory and time required for tuning these two parameters. In practice, hyper-parameter tuning can not be avoided as we have shown that the optimal trade-off between p_f and p_s was problem dependent. It would therefore be interesting to design an efficient strategy to automatically find and adjust the values of p_s and p_f, taking into account the global memory constraint. Our simplistic framework also only accounts for the memory required to store the training set in memory and not for the total memory required to actually build the ensemble.

We have only explored uniform sampling of patches of fixed size in our experiments. In the context of the Pasting approach, Breiman proposed an iterative instance weighting scheme that proved to be more efficient than uniform sampling [1]. It would be interesting to extend this approach when sampling both

instances and features. Yet, parallelization would not be trivial anymore, although probably still possible in the line of the work in [7].

Finally, our analysis of RP is mostly empirical. In the future, we would like to strengthen these results with a more theoretical analysis. A starting point could be the work in [13] that studies a scheme similar to the Pasting method applied to linear models trained through parallel stochastic gradient descent. The extension of this work to non parametric tree-based estimators does not appear trivial however, since these latter are not well characterized theoretically.

Acknowledgements. The authors would like to thank Raphaël Marée and the reviewers for their helpful feedback. GL and PG are respectively research fellow and research associate of the FNRS, Belgium. This work is supported by PASCAL2 and the IUAP DYSCO, initiated by the Belgian State, Science Policy Office.

References

1. Breiman, L.: Pasting small votes for classification in large databases and on-line. Machine Learning 36(1), 85–103 (1999)
2. Ho, T.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
3. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees (1984)
4. Breiman, L.: Bagging predictors. Machine learning 24(2), 123–140 (1996)
5. Breiman, L.: Random forests. Machine learning 45(1), 5–32 (2001)
6. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning 63(1), 3–42 (2006)
7. Chawla, N.V., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P.: Learning ensembles from bites: A scalable and accurate approach. J. Mach. Learn. Res. 5, 421–451 (2004)
8. Basilico, J., Munson, M., Kolda, T., Dixon, K., Kegelmeyer, W.: Comet: A recipe for learning and using large ensembles on massive data. In: IEEE 11th International Conference on Data Mining (ICDM), pp. 41–50. IEEE (2011)
9. Panov, P., Džeroski, S.: Combining Bagging and Random Subspaces to Create Better Ensembles. In: Berthold, M., Shawe-Taylor, J., Lavrač, N. (eds.) IDA 2007. LNCS, vol. 4723, pp. 118–129. Springer, Heidelberg (2007)
10. Pedregosa, F., et al.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
11. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
12. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research 7, 1–30 (2006)
13. Zinkevich, M., Weimer, M., Smola, A., Li, L.: Parallelized stochastic gradient descent. In: Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R., Culotta, A. (eds.) Advances in Neural Information Processing Systems, vol. 23, pp. 2595–2603 (2010)

An Efficiently Computable Support Measure for Frequent Subgraph Pattern Mining

Yuyi Wang and Jan Ramon

Department of Computer Science
Katholieke Universiteit Leuven, Heverlee 3001, Belgium
{yuyi.wang,jan.ramon}@cs.kuleuven.be

Abstract. Graph support measures are functions measuring how frequently a given subgraph pattern occurs in a given database graph. An important class of support measures relies on overlap graphs. A major advantage of the overlap graph based approaches is that they combine anti-monotonicity with counting occurrences of a pattern which are independent according to certain criteria. However, existing overlap graph based support measures are expensive to compute.

In this paper, we propose a new support measure which is based on a new notion of independence. We show that our measure is the solution to a linear program which is usually sparse, and using interior point methods can be computed efficiently. We show experimentally that for large networks, in contrast to earlier overlap graph based proposals, pattern mining based on our support measure is feasible.

Keywords: Graph mining, frequent subgraph pattern mining, support measure, frequency counting, overlap graph, linear program.

1 Introduction

Graph mining is a subfield of structured data mining. An important task is *frequent subgraph pattern mining*, which concerns the problem of finding subgraph patterns that occur frequently in a collection of graphs or in a single large graph. In this paper, we consider the single-graph setting, and we will call the large graph containing all data the *database graph*. Referring to many applications, such as social networks, the Internet, chemical and biological interaction networks, traffic networks and citation networks, the database graph is also often called the *network*.

In order to define a frequent pattern mining problem precisely, a *support measure* (also called *frequency measure*) is needed. In the problem setting where patterns are mined in a set of transactions (e.g., itemset mining [1]), a simple support measure is to count the number of transactions in which the pattern occurs. However, in the context of a single large graph, the issue is less straightforward and several articles have considered this issue [2,4,5,6].

An important drawback of the strategy to just use the number of occurrences of a pattern (either embeddings or images) as its support is that it is not *anti-monotonic*, i.e., the support of a pattern may be larger than the support of

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 362–377, 2012.

one of its subpatterns. The anti-monotonicity of the support measure (or more generally interestingness measure) plays a very important role in the design of a pattern miner, as it allows for pruning the search space [7]. Nevertheless, anti-monotonicity alone is not enough. For example, a support measure just returning a constant is anti-monotonic, but not informative. From a statistical point of view, the value of a set of examples increases if these examples are more independent. Calders et al. [6] proposed to use the situation where occurrences of a subgraph pattern are independent (i.e., they do not overlap according to some notion of overlap) as a reference. In particular, the notion of a normalized graph support measure was defined: a support measure is *normalized* if for every pattern which has only non-overlapping occurrences in a database graph, its support in that database graph equals the number of occurrences.

An important class of support measures relies on *overlap graphs*. The vertices in an overlap graph represent occurrences of a given pattern, and two vertices are adjacent iff the corresponding occurrences overlap in the database graph (according to some notion of overlap, such as sharing a vertex or an edge). An overlap graph therefore summarizes how many times a pattern occurs in the database graph, and how independent these occurrences are. An overlap graph based support measure (OGSM) takes an overlap graph of a pattern in a database graph as its input, and outputs the support of that pattern in that database graph. Vanetik et al. [2] proposed the MIS measure, the size of the maximum independent set of the overlap graph. This is intuitively appealing since it measures how often we observed a pattern occurring independently. Unfortunately, computing the MIS of an overlap graph is NP-hard [8], and remains so even for bounded degree graphs. Moreover, it has been shown that MIS cannot be approximated even within a factor of $n^{1-o(1)}$ in polynomial time unless P=NP [9], where n is the order of the overlap graph. Calders et al. [6] proposed the Lovász theta function ϑ (see e.g., [10,11]), which is computable in time polynomial in the order of the overlap graph using semidefinite programming (SDP). A straightforward application of a general purpose SDP solver yields a running time of $O(n^{6.5})$ [17]. An SDP primal-dual algorithm for approximating ϑ with a multiplicative error of $(1+\epsilon)$ was proposed [12], and the running time of this algorithm is $O(\epsilon^{-2}n^5 \log n)$. Iyrngar et al. [15] considered subgradient methods for approximating ϑ, which run in time $O(\epsilon^{-2}\log^3(\epsilon^{-1})n^4 \log n)$ in the worst case. Unfortunately, even these approximative methods are still computationally too expensive for our purposes.

In this paper, we propose a new support measure `s` that is based on bounding the value of all occurrences of a pattern that share a particular part of the database graph, and `s` can be computed efficiently using a linear program (LP). The measure `s` is not a traditional OGSM, because its output does not depend only on the overlap graph considered in earlier papers. We introduce the notion *overlap hypergraph*, and `s` is an overlap hypergraph based support measure (OHSM). We prove that `s` is anti-monotonic and normalized. Furthermore, we show that all normalized anti-monotonic OHSMs are bounded. Our empirical analysis shows that this idea yields the first support measure which is

both overlap based (and hence appealing from a statistical point of view) and computationally feasible.

The remainder of this paper is structured as follows. In the next section, we briefly review some basic notation from graph theory and formalize support measures, overlap graphs and overlap hypergraphs. In Section 3, we introduce the new measure `s` and model it as an LP. We prove that `s` is normalized and anti-monotonic in Section 4. The property that all normalized anti-monotonic OHSMs are bounded is shown in Section 5. Section 6 points out a phase transition phenomenon between frequent and infrequent patterns. Section 7 presents experimental results. Section 8 concludes the paper with an overview of our contributions.

2 Preliminaries

2.1 Graph Theory

We recall basic graph theoretic notions used in this paper. For more background in this area, see also [13].

Graphs. A *graph* G is an ordered pair (V, E), where V is a set of *vertices* and E is either a set of *edges* $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ or a set of *arcs* $E \subseteq \{(u, v) \mid u, v \in V, u \neq v\}$. In the former (latter) case, we call the graph *undirected* (*directed*). Vertices are *adjacent* if there is an edge (arc) between them. For an edge $e = \{u, v\}$ (arc $e = (u, v)$), u and v are *incident* with e.

A *labeled graph* is a quadruple $G = (V, E, \Sigma, \lambda)$, with (V, E) a graph, Σ a non-empty finite set of labels, and λ a function assigning labels in Σ to the vertices or edges (or arcs), or both. For simplicity, by labeled graph, we will mean *vertex-labeled graph* unless explicitly pointed out.

We will use the notation $V(G)$, $E(G)$ and λ_G to refer to the set of vertices, the set of edges (or arcs) and the labeling function of a graph G, respectively. g is said to be a *subgraph* of G if $V(g) \subseteq V(G)$, $E(g) \subseteq E(G)$ and for all $v \in V(g)$ that $\lambda_g(v) = \lambda_G(v)$, and write $g \subseteq G$.

We denote $\mathcal{G}$ the class of all graphs, and $\mathcal{G}^{\leftrightarrow}$ ($\mathcal{G}^{\rightarrow}$), the restriction to undirected (directed) graphs, while $\mathcal{G}_{\lambda}$ ($\mathcal{G}_{\bullet}$) denotes the restriction to labeled (unlabeled) graphs. One can combine notations, e.g., $\mathcal{G}_{\bullet}^{\rightarrow}$ for the class of directed, unlabeled graphs.

An *independent set* I of $G \in \mathcal{G}$ is a subset of $V(G)$ such that no pair of distinct vertices of I is adjacent in G. A *clique* Q of $G \in \mathcal{G}$ is a subset of $V(G)$ such that for all distinct vertices $v, w \in Q$, u and v are adjacent in G. A *clique partition* $\Pi = \{s_1, s_2, \cdots, s_k\}$ of $G \in \mathcal{G}$ is a partition of $V(G)$ such that every set s in Π is a clique.

Morphisms. The following concepts defined in terms of $\mathcal{G}_{\lambda}^{\rightarrow}$ are also valid for undirected and/or unlabeled graphs by dropping the direction of the edges and/or the labels of the vertices.

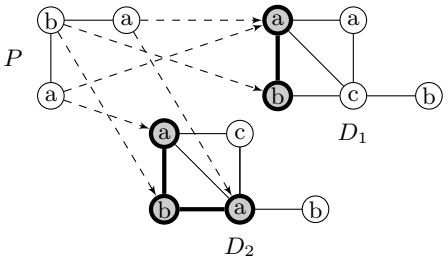


Fig. 1. Homomorphism and isomorphism. A homo-image (but not iso-image) of P is highlighted in D_1, and an iso-image of P is highlighted in D_2.

A *homomorphism* ψ from $G \in \mathcal{G}_\lambda^\rightarrow$ to $G' \in \mathcal{G}_\lambda^\rightarrow$ is a mapping from $V(G)$ to $V(G')$ such that for all $v \in V(G) : \lambda_G(v) = \lambda_{G'}(\psi(v))$ and for all $(u, v) \in E(G) : (\psi(u), \psi(v)) \in E(G')$. We call ψ *vertex-surjective* if $\forall v' \in V(G') : \exists v \in V(G) : \psi(v) = v'$, and call it *edge-surjective* if $\forall (u', v') \in E(G') : \exists (u, v) \in E(G) : \psi(u) = u'$ and $\psi(v) = v'$. A homomorphism is *surjective* if it is both vertex- and edge-surjective.

An *isomorphism* from $G \in \mathcal{G}_\lambda^\rightarrow$ to $G' \in \mathcal{G}_\lambda^\rightarrow$ is a bijective homomorphism ψ from G to G'. In this case, we say that G is *isomorphic* to G' and write $G \cong G'$. We use $G \subseteq G'$ to denote that $G \cong g$, for some subgraph g of G'. This is equivalent to saying that there exists a *subgraph isomorphism* from G to G'.

An *iso-image* (*homo-image*) g of $P \in \mathcal{G}_\lambda^\rightarrow$ in $D \in \mathcal{G}_\lambda^\rightarrow$ is a subgraph $g \subseteq D$ for which there exists an isomorphism (surjective homomorphism) ψ from P to g. We call g the iso-image (homo-image) through ψ. An individual isomorphism (homomorphism) ψ from P to g is called an *iso-embedding* (*homo-embedding*) of P in D. See Fig. 1 for an example.

In this paper, we only consider iso-images, although the measure `s` can be generalized for other matching operators such as homomorphism. We use the term *image* instead of iso-image afterwards, and denote with $\mathrm{Img}(D, P)$ the set of all images of P in D. Suppose $g \in \mathrm{Img}(D, p)$ and $g' \in \mathrm{Img}(D, P)$, if g is a subgraph of g', we call g a *subimage* of g' and g' a *superimage* of g.

Hypergraphs. A *hypergraph* is an ordered pair (V, E), where V is a set of vertices and E is a set of *hyperedges* $E \subseteq 2^V$. We denote $\mathcal{H}$ the class of all hypergraphs. As in the case of graphs, for $H \in \mathcal{H}$, $V(H)$ denotes the set of vertices and $E(H)$ denotes the set of hyperedges. To every hypergraph H which has n vertices and m hyperedges, we associate an $n \times m$ *incidence matrix* $M_H = (m_{ij})$ where $m_{ij} = 1$ if $v_i \in e_j$ and $m_{ij} = 0$ otherwise.

2.2 Support Measures

We review the concepts and properties of support measures and overlap graphs, and introduce the new concept of overlap hypergraphs.

Definition 1. *A* support measure *is a function* $f : \mathcal{G} \times \mathcal{G} \mapsto \mathbb{R}$ *that maps* (D, P) *to a non-negative number* $f(D, P)$*, where* P *is called the pattern,* D *the database graph and* $f(D, P)$ *the* support *of* P *in* D*.*

For efficiency reasons, most graph miners generate patterns from smaller patterns to larger ones [14]. Such a method requires the support measure to be anti-monotonic.

Definition 2. *A support measure* f *is* anti-monotonic *if for all* p, P, D *in* $\mathcal{G}$: $p \subseteq P \Rightarrow f(D, P) \leq f(D, p)$*.*

As explained in the introduction, anti-monotonicity alone is not enough. It is also desirable that the support measure accounts for the independence of the occurrences of the patterns. We can define *overlap* in different ways [6]. Popular definitions are *vertex-overlap*, i.e., two images g_1 and g_2 overlap if $V(g_1) \cap V(g_2) \neq \emptyset$, and *edge-overlap*, i.e., two images g_1 and g_2 overlap if $E(g_1) \cap E(g_2) \neq \emptyset$. Edge-overlap implies vertex-overlap. In this paper, by overlap, we will mean vertex-overlap, although our results are also valid in the edge-overlap setting.

Definition 3. *A support measure* f *is* normalized *if for all* P, D *in* $\mathcal{G}$: $f(D, P) = |Img(D, P)|$ *when there do not exist two distinct images* g_1 *and* g_2 *in* $Img(D, P)$ *satisfying* $V(g_1) \cap V(g_2) \neq \emptyset$*.*

Overlap Graphs. The notion of overlap graph plays an important role in the design and computation of anti-monotonic measures. Given a pattern P and a database graph D, the overlap graph of P in D is a graph $G_P^D \in \mathcal{G}_\bullet^\leftrightarrow$. Every vertex of G_P^D is an image of P in D, that is, $V(G_P^D) = \text{Img}(D, P)$. Two vertices u and v are adjacent in G_P^D if they overlap.

Vanetik et al. [3] define the induced support measure $f(G_P^D) = f(D, P)$, which we call an overlap graph based support measure (OGSM). They proposed the first normalized anti-monotonic OGSM, the size of the maximum independent set (MIS) [2]. Later, Calders et al. [6] proposed two normalized anti-monotonic OGSMs, the size of a minimum clique partition (MCP) and the Lovász theta value (ϑ). As mentioned in the introduction, these existing OGSMs are very expensive to compute.

Overlap Hypergraphs. As we are using vertex-overlap, each vertex v in a database graph D determines a clique in the overlap graph G_P^D in which P is a pattern. That is, suppose v is a vertex in D, then $Img_v(D, P) = \{g \in \text{Img}(D, P) \mid v \in V(g)\}$ build a clique in G_P^D since the images overlap at the vertex v.

We define the *overlap hypergraph* of P in D, denoted H_P^D as the hypergraph whose vertices are the images $\text{Img}(D, P)$, and for each vertex $v \in V(D)$ a hyperedge $e_v \in E(H_P^D)$ such that $e_v = \{g \in V(H_P^D) \mid v \in V(g)\}$. The hyperedges represent cliques in G_P^D.

In an overlap hypergraph H_P^D, we say that a hyperedge e is *dominated* by another hyperedge e' if $e \subset e'$, and a hyperedge e is *dominating* if it is not

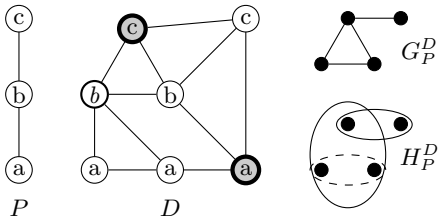


Fig. 2. Overlap graph and overlap hypergraph. Given a pattern P, a database graph D, the overlap graph G_P^D and the overlap hypergraph H_P^D are shown on the right. In the overlap hypergraph, the (dominating) hyperedges are determined by the highlighted vertices in the database graph, and a dominated hyperedge is given in a dashed ellipse.

dominated by any other hyperedge. For any D and P, we define the reduced overlap hypergraph $\tilde{H}_P^D$ to be the hypergraph for which $V(\tilde{H}_P^D) = V(H_P^D)$ and $E(\tilde{H}_P^D)$ is the set of all dominating hyperedges of H_P^D. In the sequel we only refer to $\tilde{H}_P^D$. We will abuse terminology and simply call $\tilde{H}_P^D$ the overlap hypergraph. See Fig. 2 for an example.

We henceforth refer to the induced support measure, which we denote by $f(\tilde{H}_P^D)$, instead of referring to $f(D, P)$. Such induced support measures are called overlap hypergraph based support measures (OHSM). We call OHSMs and OGSMs overlap based support measures.

3 A New Normalized Anti-monotonic Measure

We introduce a new normalized anti-monotonic OHSM, which we denote s. It satisfies the desirable properties of being anti-monotic and normalized, and can be computed efficiently.

The MIS measure is a normalized anti-monotonic OGSM. Note that given an overlap hypergraph $\tilde{H}_P^D$, we are able to derive the corresponding overlap graph G_P^D by replacing every hyperedge with a clique. Therefore, we can rephrase the definition of the MIS measure using overlap hypergraphs. Suppose $\tilde{H}_P^D$ is an overlap hypergraph:

$$MIS(\tilde{H}_P^D) = \max |\{I \subseteq V(\tilde{H}_P^D) \mid \forall e \in E(\tilde{H}_P^D) : |e \cap I| \leq 1\}| \tag{1}$$

The MIS measure requires that a vertex of an overlap (hyper)graph is either in the independent set I or not. Our new measure s is a relaxation of the MIS measure by allowing counting vertices of an overlap hypergraph partially.

Let $\tilde{H}_P^D$ be an overlap hypergraph. We start by assigning to each vertex v of $\tilde{H}_P^D$ a variable x_v. We then consider vectors $x \in \mathbb{R}^{V(\tilde{H}_P^D)}$ of variables where for every $v \in V(\tilde{H}_P^D)$, x_v denotes the variable (component of x) corresponding to v. x is *feasible* iff it satisfies

$$\begin{aligned} &\text{(i) } \forall v \in V(\tilde{H}_P^D) : 0 \leq x_v \\ &\text{(ii) } \forall e \in E(\tilde{H}_P^D) : \textstyle\sum_{v \in e} x_v \leq 1. \end{aligned}$$

We denote the feasible region (the set of all feasible $x \in \mathbb{R}^{V(\tilde{H}_P^D)}$) with $\mathfrak{R}(\tilde{H}_P^D)$. It is a convex polytope. The measure s is defined by

$$\mathsf{s}(\tilde{H}_P^D) = \max_{x \in \mathfrak{R}(\tilde{H}_P^D)} \sum_{v \in V(\tilde{H}_P^D)} x_v \tag{2}$$

Clearly, s is the solution to a linear program.

We will call an element $x \in \mathfrak{R}(\tilde{H}_P^D)$ which makes $\sum_{v \in V(\tilde{H}_P^D)} x_v$ maximal a *solution* to the LP of s.

There are very effective methods for solving LPs, including the simplex method which is efficient in practice though its complexity is exponential, and the more recent interior-point methods [16]. The interior-point method solves an LP in $O(n^2 m)$ time, where n (here $\min\{|V(\tilde{H}_P^D)|, |E(\tilde{H}_P^D)|\}$) is the number of variables, and m (here $|V(\tilde{H}_P^D)| + |E(\tilde{H}_P^D)|$) is the number of constraints. Usually, patterns are not large, so the LPs for computing s are sparse. Almost all LP solvers perform significantly better for sparse LPs.

4 Conditions for Anti-monotonicity

Vanetik et al. [3] gave necessary and sufficient conditions for anti-monotonicity of for OGSMs on labeled graph using edge-overlap. This result was generalized in [6] to any OGSM on labeled or unlabeled, directed or undirected graphs using edge overlap or vertex overlap and isomorphism, homomorphism or homeomorphism. Our conditions for anti-monotonicity are based on the overlap hypergraphs. Our main result is that an OHSM is anti-monotonic if and only if it is non-decreasing under certain operations on the overlap hypergraph.

We begin by defining three operations on any overlap hypergraph, which we will then use in our conditions for anti-monotonicity. These operations are different from those used in [3,6], but play a similar role. As mentioned in these earlier papers, the motivation for these operations is that it is often easier to show that an OHSM satisfies the conditions of the theorem (being non-decreasing under the three operation), than to show anti-monotonicity of a measure directly.

For $H \in \mathcal{H}$, we define:

- Vertex Addition: A new vertex v is added to every existing hyperedge: $VA(H, v) = (V(H) \cup \{v\}, \{e \cup \{v\} \mid e \in E(H)\})$.
- Subset Contraction: Let $K \subseteq V(H)$ be a set of vertices of the hypergraph such that $\exists e \in E(H) : K \subseteq e$. Then, the subset contraction operation contracts K into a single vertex k, which remains in only those hyperedges that are supersets of K. Formally, $SC(H, K, k) = (V(H) - K \cup \{k\}, E_1 \cup E_2)$ where $E_1 = \{e - K \cup \{k\} \mid e \in E(H) \text{ and } K \subseteq e\}$ and $E_2 = \{e - K \mid e \in E(H) \text{ and } K \nsubseteq e\}$).
- Hyperedge Split: This operation splits a size k hyperedge into k hyperedges of size $(k-1)$ each: $HS(H, e) = (V(H), E(H) - \{e\} \cup \{e - \{v\} \mid v \in e\})$, where $e \in E(H)$.

For example, suppose H_0 is a hypergraph, $V(H_0) = \{v_1, v_2, v_3, v_4\}$, and $E(H_0)$ contains two hyperedges $\{v_1, v_2, v_3\}$ and $\{v_1, v_4\}$. Let $H_1 = VA(H_0, v_5)$, then $V(H_1) = \{v_1, v_2, v_3, v_4, v_5\}$ and $E(H_1)$ contains hyperedges $\{v_1, v_2, v_3, v_5\}$ and $\{v_1, v_4, v_5\}$. Let $H_2 = SC(H_1, \{v_1, v_3\}, v_6)$, then $V(H_2) = \{v_2, v_4, v_5, v_6\}$ and $E(H_2)$ contains hyperedges $\{v_2, v_5, v_6\}$ and $\{v_4, v_5\}$. Let $H_3 = HS(H_2, \{v_2, v_5, v_6\})$, then $V(H_3) = V(H_2)$ and $E(H_2)$ contains four hyperedges $\{v_2, v_5\}$,$\{v_2, v_6\}$, $\{v_5, v_6\}$ and $\{v_4, v_5\}$.

4.1 Sufficient Condition

We give a sufficient condition for support measure anti-monotonicity in terms of the three operations on the overlap hypergraph that we have defined.

Theorem 1. *Let $f' : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ be a support measure, and $f : \mathcal{H} \to \mathbb{R}$ with $f'(D, P) = f(\tilde{H}_P^D))$ be the induced OHSM. If f is non-decreasing under VA, SC and HS, then f' is an anti-monotonic support measure.*

Proof. Suppose D is a database graph, and p and P are two patterns such that p is a subgraph of P. We prove that $\tilde{H}_p^D$ can be obtained from $\tilde{H}_P^D$ by applying only the operations VA, SC and HS. It follows then that $f'(D, P) = f(\tilde{H}_P^D) \leq f(\tilde{H}_p^D) = f'(D, p)$ for any D, P and p, proving the theorem.

Let $<$ be an arbitrary order defined on $V(\tilde{H}_p^D)$. We define for $v \in V(\tilde{H}_p^D)$ the set $\Pi_v = \{u \in V(\tilde{H}_P^D) \mid v \preceq u \text{ and } \forall w < v : w \not\preceq u\}$. Here, remember that the vertices of $\tilde{H}_p^D$ are images of p and hence $v \preceq u$ refers to a subgraph isomorphism relationship between v and u.

The Π_v are pairwise disjoint and $\cup_{v \in V(\tilde{H}_p^D)} \Pi_v = V(\tilde{H}_P^D)$. We point out that there may exist vertices v for which $\Pi_v = \emptyset$. We divide $V(\tilde{H}_p^D)$ into two sets $V_0 = \{v \mid \Pi_v = \emptyset\}$ and $V_1 = \{v \mid \Pi_v \neq \emptyset\}$.

Let H be a hypergraph initially equal to $\tilde{H}_P^D$. We will perform operations VA, SC and HS on H, until finally it is equal to $\tilde{H}_p^D$.

First, H is modified by a sequence of VA operations. For each $v \in V_0$, we do $H := VA(H, v)$. Now, $\forall e \in E : V_0 \subseteq e$.

Then, for each $v \in V_1$, we perform $H := SC(H, \Pi_v, v)$. The operations are valid because for $v \in V_1$ each vertex $u \in \Pi_v$ stands for a superimage of the same v, i.e., $v \preceq u$ and hence $\exists e \in E(H) : \Pi_v \subseteq e$. It is easy to verify that now $V(\tilde{H}_p^D) = V(H)$ holds.

Consider a hyperedge $e'_x \in E(\tilde{H}_p^D)$ which is determined by $x \in V(D)$, i.e., $e'_x = \{v \in V(\tilde{H}_p^D) \mid x \in V(v)\}$. We know that $e'_x \cap V_0$ is a subset of any $e \in E(H)$. $E(\tilde{H}_P^D)$ has a dominating hyperedge e''_x determined by x, i.e., $e''_x = \{v \in V(\tilde{H}_P^D) \mid x \in V(v)\}$ (or has another hyperedge e''_y which is a superset of the dominated hyperedge e''_x). We have $e'_x \subseteq e''_x$ (or e''_y). Thus, $\forall v \in e'_x \cap V_1 : \Pi_v \subseteq e''_x$ (or e''_y). Therefore, there must be a hyperedge $e \in E(H)$ such that $e'_x \subseteq e$. This property shows that every hyperedge in $E(\tilde{H}_p^D)$ either exists in $E(H)$ or can be obtained later on by performing a sequence of HS on H. □

Theorem 2. $\mathsf{s}(D, P) = \mathsf{s}(\tilde{H}_P^D)$ *is a normalized anti-monotonic support measure.*

Proof. First, we prove s is normalized. If the pattern P only has non-overlapping images in the database graph D, every hyperedge in $E(\tilde{H}_P^D)$ contains only one vertex, then setting $x_v = 1$ for every $v \in V(\tilde{H}_P^D)$ is a feasible assignment and is clearly maximal. That is, s equals the number of non-overlapping images. Therefore, s is normalized.

Then, we prove s is anti-monotonic using Theorem 4.1. Suppose H is an overlap hypergraph and x^* is a solution to the LP of $\mathsf{s}(H)$. Let H_1 be the overlap hypergraph $VA(H, v)$, and let $x_u = x_u^*$ for all vertices $u \neq v$ and $x_v = 0$. x is a feasible solution for the LP of $\mathsf{s}(H_1)$, so $\mathsf{s}(H_1) \geq \sum_v x_v = \mathsf{s}(H)$. Let H_2 be the overlap hypergraph $SC(H, K, k)$, and let $x_u = x_u^*$ for all vertices $u \neq k$ and $x_k = \sum_{v \in K} x_v^*$. x is a feasible for the LP of $\mathsf{s}(H_2)$, so $\mathsf{s}(H_2) \geq \sum_v x_v = \mathsf{s}(H)$. Let H_3 be the overlap hypergraph $HS(H, e)$. x^* is also a feasible for the LP of $\mathsf{s}(H_3)$, so $\mathsf{s}(H_3) \geq \mathsf{s}(H)$. □

4.2 Necessary Condition

We show that the above sufficient condition for anti-monotonicity is also necessary.

Theorem 3. *Let $f' : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ be a support measure, and $f : \mathcal{H} \to \mathbb{R}$ with $f'(D, P) = f(\tilde{H}_P^D))$ be the induced OHSM. If f' is anti-monotonic, then f is non-decreasing under VA, SC and HS.*

Proof (sketch). Let H_P be any hypergraph and H_p a hypergraph obtained by performing VA, SC or HS on H_P. We show that there exists a database graph D and patterns P and p such that $\tilde{H}_P^D = H_P$ and $\tilde{H}_p^D = H_p$. Then, there follows $f(H_P) = f'(D, P) \leq f'(D, p) = f(H_p)$ which proves the theorem. For convenience, we show the theorem only for $D, P, p \in \mathcal{G}_\lambda^\leftrightarrow$, but the proof can be generalized.

In Figure 3, we give the patterns P and p ($p \subseteq P$), and list different types of overlap. The numbers of vertices with label a or b in P and p are not fixed, and we can assume that P and p have enough such vertices. We construct database graphs by combining the patterns using these different types of overlap. We name the different types O_1, O_2, O_3 and O_4. In Figure 3, only two patterns overlap for each type, but during the construction of database graphs, it is allowed that more than two patterns overlap at the same vertex.

If $\tilde{H}_p^D = VA(\tilde{H}_P^D, v)$, then we can construct the database graph using O_1 and O_2. O_1 is used to determine all the hyperedges in $E(\tilde{H}_P^D)$. O_2 is used to introduce a new vertex and make the new vertex exist in every hyperedge in $E(\tilde{H}_p^D)$.

If $\tilde{H}_p^D = SC(\tilde{H}_P^D, K, k)$, then we can construct the database graph using O_1, O_3 and O_4. O_3 is used to build the subset K. O_4 is used to determine the hyperedges $e \in E(\tilde{H}_P^D)$ which satisfy $e \cap K \neq \emptyset$ and $K \nsubseteq e$. For any hyperedge e determined by O_4, there is a hyperedge $e' \in E(\tilde{H}_p^D)$ determined by O_1 such that $e' = e - K$. Besides, O_1 also determines all hyperedges $e \in \tilde{H}_P^D$ which satisfy $K \subseteq e$ or $e \cap K = \emptyset$.

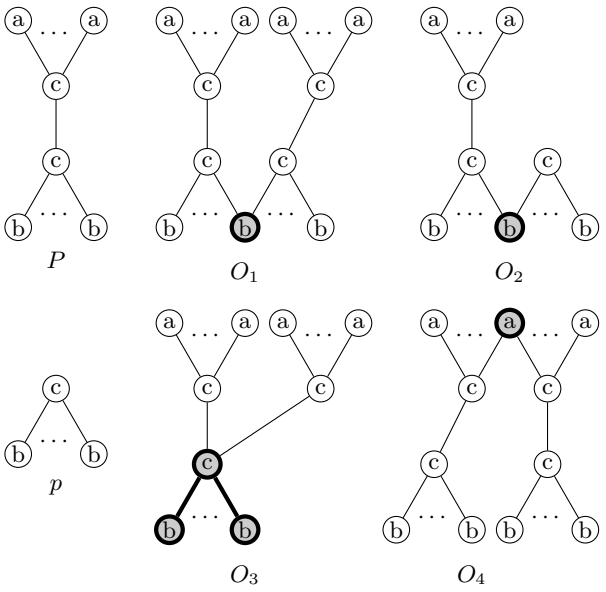


Fig. 3. Patterns and different types of overlap. The highlighted parts show the ways two patterns overlap.

If $\tilde{H}_p^D = HS(\tilde{H}_P^D, e)$, then we can construct the database graph using O_1 and O_4. O_4 is used to build the hyperedge e. O_1 determines the hyperedges $\{e - \{v\} \mid vn \in e\} \in E(\tilde{H}_p^D)$ and all other hyperedges. □

5 Bounding Theorem

In [6], the authors showed an interesting result that all normalized anti-monotonic OGSMs are bounded (between the maximum independent set size (MIS) and the minimum clique partition size (MCP)). Similarly, we prove that all normalized anti-monotonic OHSMs are also bounded. We first introduce another OHSM on $H \in \mathcal{H}$, the size of a minimum set cover of H:

$$MSC(H) = \min |\{S \subseteq E(H) \mid \bigcup_{e \in S} e = V(H)\}| \tag{3}$$

It is not difficult to verify that MSC is normalized and anti-monotonic. To compute MSC is an NP-hard problem. The maximum independent set size (Eq. (1)) and minimum vertex cover (Eq. (3)) are the minimal and the maximal possible normalized anti-monotonic OHSMs.

Theorem 4. *For every normalized anti-monotonic OHSM f, and every $H \in \mathcal{H}$, it holds that: $MIS(H) \leq f(H) \leq MSC(H)$.*

Proof. We use Theorem 3 to show the minimality of MIS and the maximality of MCP, respectively.

Let H be a hypergraph, and let $I = \{v_1, v_2, \cdots, v_k\}$ be a maximum independent set of H. Starting from the hypergraph $H_I = (\{v_1, v_2, \cdots, v_k\}, \{\{v_1\}, \{v_2\}, \cdots, \{v_k\}\})$, we can get H by adding vertices $V(H) - I$ using VA first and then spliting hyperedges by a sequence of HS. Since f is normalized, it is anti-monotonic and therefore f cannot decrease after each step, and $f(H_I) = k$. As such, $f(H)$ is larger than or equal to $k = MIS(H)$.

On the other hand, let $\{e_1, e_2, \cdots, e_k\}$ be a minimum set cover for H and let $H_{sc} = SC(\ldots SC(SC(H, e_1, v_{e_1}), e_2, v_{e_2}) \cdots, e_k, v_{e_k})$. H_{sc} only has the hyperedges with exact one vertex in each of them. Because f is anti-monotonic, f is not decreasing under SC and thus $f(H) \leq f(SC(H, e_1)) \leq \cdots \leq f(H_{sc}) = k$.

□

6 The Phase Transition from Frequent to Infrequent

Large real-world networks are known to satisfy properties similar to random graphs. A well-known property is that properties which can be expressed in first order logic are satisfied by either almost all graphs or almost no graphs (0-1 law, see [21]). For random graphs, one can observe (see also our experiments below) that for a given pattern P, it is either very easy to embed the pattern in the network, or very difficult. This leads to another 0-1 property: the frequency of many patterns is either very low or very high (for our s measure, nearly equal to the network size). Consider e.g. a social network and the pattern "X is a friend of Y and Y is a friend of Z". Since most people have at least two friends, such pattern will match about everywhere. This holds more generally for many tree and path patterns. In fact, most such patterns are overly general and not very interesting.

In the context of overlap-graph based support measures, these overly general patterns also pose a computational problem: since they match about everywhere, the corresponding overlap graph is very large. Therefore, for these less interesting overly general patterns, our prototype implementation just records that they are very frequent but doesn't attempt to compute their frequency exactly by constructing the overlap graph explicitely. We hence distinct three categories of patterns: the infrequent patterns, the moderately frequent patterns, and the very frequent patterns (for which the frequency will not be computed exactly).

7 Experiments

This section provides experimental results, illustrating the practical potential of our new measure s.

7.1 Experimental Setup

For our experiments, we are interested in answering the following experimental questions:

Q1 How does the computational cost of the s measure compare to other existing overlap based support measures, e.g., Lovász ϑ value?
Q2 How does the cost of computing the s measure compare to the cost of listing the embeddings?
Q3 Is it feasible to mine all s-frequent patterns of size up to 6 in moderately sized networks?
Q4 What can we learn about the phase transition between frequent and infrequent and the randomness of the DBLP dataset?

7.2 Results

All experiments are run on an Intel Core i7-2600 CPU (3.4Gz) with 8Gb RAM.

We use the algorithm VF2 (implemented in C++) to find embeddings of patterns in networks [19]. We use Matlab 2012a and SeduMi 1.21 to solve the LPs for the s measure and the SDPs for the Lovász ϑ value. The desired accuracy of all LPs and SDPs is 10^{-4}.

Lovász ϑ Function. In the first experiment, we generate hypergraphs randomly, and convert them into graphs by replacing the hyperedges with cliques. The hypergraphs are used to compute s measures, while the graphs are used to compute the Lovász ϑ measure. The hypergraphs have $20, 40, \ldots, 200$ vertices and $20, 40, \ldots, 100$ hyperedges. With probability 0.05, a vertex of the hypergraphs appears in a hyperedge.

Fig. 4 shows the time cost to compute the s measure and the Lovász ϑ measure for these graphs. θ_m and s_m means there are m hyperedges.

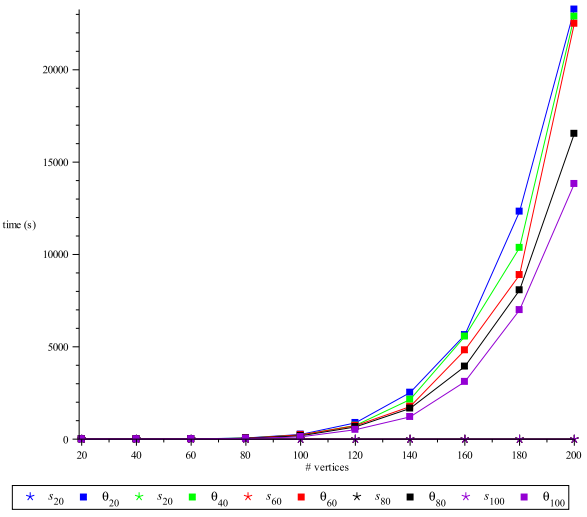


Fig. 4. Time consumed to compute θ and s

Real-World Data. We use two DBLP co-authorship networks (DBLP0305 showing co-authorships from 2003 to 2005 and DBLP0507 showing co-authorships from 2005 to 2007) [18]. If an author i co-authored a paper with author j, the networks contain an undirected edge $\{i, j\}$. The vertices are unlabeled, whereas the edges are labeled with an integer indicating the year the edge first appeared in. The network dblp0305 has 109944 vertices, 228461 edges and 3 different labels. The network dblp0507 has 135516 vertices, 290363 edges and 3 different labels. In this experiment, we choose 1.5% as the frequency threshold.

For each network, we start from patterns of level 1, the single vertices. A pattern which has $i-1$ edges is a *candidate* in level i ($i \geq 2$) if none of its subpatterns is infrequent and at least one of its subpatterns in level $i-1$ is frequent (others may have too many embeddings). If a pattern has more than 5.10^6 embeddings, we don't compute s but can easily show that the pattern is frequent. We call such a pattern *very frequent*.

Table 1 gives the results of the experiments that mine frequent patterns up to level 6 in the DBLP networks. T_{map} is the average time per pattern to find embeddings using VF2, and T_{s} is the average time to compute s. Both are in seconds.

Synthetic Data. We generate scale-free networks of different sizes [20]. They have $10^2, 10^3, \ldots, 10^6$ vertices which are labeled by 4 different labels, and all of them have the same average degree 10. We will call them 10_X networks, where $X = 2, 3, 4, 5, 6$. In this experiment, all tree patterns are very frequent. Therefore, we only report statistics for the non-tree patterns. We choose the frequency threshold 0.1%.

Tables (2)-(4) give the results of the experiments that mining frequent non-tree patterns up to level 6 (except the network which has 10^6 vertices) in the

Table 1. Frequent pattern mining in DBLP0305 and DBLP0507. Lev. = level (pattern size), Cand. = # candidate patterns, Comp. = # patterns for which s was computed, Freq. = # frequent patterns

Lev.	Cand.	Comp.	Freq.	T_{map}	T_{s}	Cand.	Comp.	Freq.	T_{map}	T_{s}
1	1	1	1	0.452	0.000251	1	1	1	0.711	0.000303
2	3	3	3	10.783	2.041	3	3	3	11.225	2.743
3	6	6	6	24.166	8.022	6	6	6	37.152	22.245
4	34	28	19	92.035	41.557	34	28	22	73.531	77.161
5	95	25	8	634.099	42.234	118	54	25	814.156	138.145
6	56	13	7	817.789	91.018	179	35	12	1530.608	430.637

Table 2. Frequent non-tree pattern mining in the 10_2 network

Level	Candidates	Computed	Frequent	T_{map}	T_{s}
4	20	20	16	0.014	0.383
5	191	191	182	0.015	0.388
6	2083	2083	2033	0.018	0.394

Table 3. Frequent non-tree pattern mining in the 10_3 and 10_4 network

Lev.	Cand.	Comp.	Freq.	T_{map}	T_{s}	Cand.	Comp.	Freq.	T_{map}	T_{s}
4	20	20	20	0.018	0.383	20	20	20	0.085	0.393
5	215	215	215	0.031	0.431	215	215	215	0.277	0.406
6	2430	2430	2422	0.128	0.481	2430	2430	2349	3.141	1.877

Table 4. Frequent non-tree pattern mining in the 10_5 and 10_6 networks

Lev.	Cand.	Comp.	Freq.	T_{map}	T_{s}	Cand.	Comp.	Freq.	T_{map}	T_{s}
4	20	20	5	2.906	0.301	20	20	0	216.196	0.428
5	99	99	9	12.435	0.673	55	55	12	1565.134	0.996
6	758	742	648	354.194	24.408	-	-	-	-	-

scale-free networks. Levels 1 to 3 only contain tree patterns, so we do not list them in the tables.

7.3 Discussion

Based on the results presented above, we can answer the experimental questions as follows:

Q1 One can see from Table 4 that, for all the randomly generated (hyper)graphs, s can be computed in a very short period of time (< 0.01 seconds), while the time consumed to compute ϑ grows fast when the number of vertices increases. Clearly, for larger (hyper)graphs on which s measure can be computed efficientlly, it is extremely difficult to compute the ϑ value in a reasonable time period by solving the corresponding SDP using existing methods. Therefore, s outperforms ϑ value in terms of efficiency.

Q2 On the real-world data, the time needed to compute embeddings is significantly larger than the time needed to compute s. For the larger synthetic datasets and the larger patterns the difference is even several orders of magnitude.

Q3 We can see that using VF2 and the s measure, frequent patterns of level up to 6 can be mined in a reasonable amount of time. In contrast to earlier approaches using the MIS or ϑ measures, here the bottleneck is clearly the pattern matching part of the algorithm. If this part can be improved, it can be expected that larger patterns can be mined in larger networks.

Q4 For the synthetic data, we found that the frequency of cyclic (non-tree) patterns was rather low, we needed a frequency threshold of 0.1% to mine them. One can conclude that while in standard random graph models nodes choose their neighbors randomly, in real-world data the connections of candidate neighbors have an important influence.

8 Conclusions

In this paper, we studied the problem of measuring how frequently a given pattern occurs in a given database graph. We have proposed a new overlap based

support measure s. In contrast to existing overlap based support measures, it can be computed efficiently. We have shown that it is anti-monotonic and normalized. The experimental results demonstrate that it is a practical overlap based measure and it is effective to prune the search space.

Compared to non-overlap based measures, e.g., the min-image support measure [4], the s measure has statistical advantages. For example, consider the embeddings: $\langle 1, 11\rangle$, $\langle 2, 11\rangle$, $\langle 3, 11\rangle$, $\langle 4, 11\rangle$, $\langle 5, 11\rangle$, $\langle 6, 12\rangle$, $\langle 6, 13\rangle$, $\langle 6, 14\rangle$, $\langle 6, 15\rangle$ and $\langle 6, 16\rangle$. Then min-image returns 6 while s returns 2. The latter equals the number of independent embeddings. Therefore, from a statistical point of view, for counting the number of independent observations of some phenomenon s is preferable.

This aim to measure only independent occurrences is shared with the MIS measure [3]. MIS is NP-hard while s is an efficiently computable relaxation. MIS returns an integer and is more strict in the sense that it never accounts for overlapping occurrences, while s also partially counts observations not explained by vertices of already counted embeddings. E.g. consider the embeddings $\langle a, b, c\rangle$, $\langle a, d, e\rangle$ and $\langle f, b, e\rangle$. The MIS is 1. However, even though each of the vertices a, b and e could have 'caused' two embeddings, no vertex is involved in all three embeddings. Therefore, s partially counts the third embedding, in this case resulting in the value 1.5.

Our proposed measure is flexible, in the sense that it is possible for a user to plug in his own definition of overlap. Investigating this in more detail is one possible line of future research. Our proposal makes measuring the frequency of a pattern in a more sound statistical way tractable. There are however other challenges related to pattern mining in networks. The major one in our experiments was the pattern matching. However, we anticipate that here too we can get a long way in making things tractable. In particular we intend to integrate our approach with recent results concerning efficient pattern matching operators based on arithmetic circuits [22].

Acknowledgements. This work was supported by ERC Starting Grant 240186 "MiGraNT: Mining Graphs and Networks: a Theory-based appraoch".

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of SIGMOD 1993, pp. 207–216 (1993)
2. Vanetik, N., Gudes, E., Shimony, S.E.: Computing frequent graph patterns from semistructured data. In: Proceeding of ICDM 2002, pp. 458–465 (2002)
3. Vanetik, N., Shimony, S.E., Gudes, E.: Support measures for graph data. Data Min. Knowl. Discov. 13(2), 243–260 (2006)
4. Bringmann, B., Nijssen, S.: What Is Frequent in a Single Graph? In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 858–863. Springer, Heidelberg (2008)
5. Fiedler, M., Borgelt, C.: Support Computation for Mining Frequent Subgraphs in a Single Graph. In: Proceedings of MLG 2007 (2007)

6. Calders, T., Ramon, J., Dyck, D.V.: All normalized anti-monotonic overlap graph measures are bounded. Data Min. Knowl. Discov. 23(3), 503–548 (2011)
7. Kuramochi, M., Karypis, G.: Finding frequent patterns in a large sparse graph. Data Min. Knowl. Discov. 11(3), 243–271 (2005)
8. Garey, M.R., Johnson, D.S.: Computers and intractibility, a guide to the theory of NP-Completeness. W. H. Freeman and Company (1979)
9. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Approximating clique is almost NP-Complete. In: FOCS, pp. 2–12. IEEE Computer Society (1991)
10. Lovász, L.: On the Shannon capacity of a graph. IEEE Transactions on Information Theory 25(1), 1–7 (1979)
11. Knuth, D.E.: The sandwich theorem. Electr. J. Comb. 1, 1–48 (1994)
12. Chan, T., Chang, K.L., Raman, R.: An SDP primal-dual algorithm for approximating the Lovsz-theta function. In: Proceedings of the IEEE ISIT 2009, pp. 2808–2812 (2009)
13. Diestel, R.: Graph theory. Springer (2010)
14. Chakrabarti, D., Faloutsos, C.: Graph mining: laws, generators, and algorithms. ACM Comput. Surv. 38(1), 1–69 (2006)
15. Iyengar, G., Phillips, D.J., Stein, C.: Approximating semidefinite packing programs. SIAM Journal on Optimization 21(1), 231–268 (2011)
16. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge Univ. Press (2004)
17. Klein, P.N., Lu, H.: Efficient approximation algorithms for semidefinite programs arising from MAX CUT and COLORING. In: Proc. of ACM STOC 1996, pp. 338–347 (1996)
18. Berlingerio, M., Bonchi, F., Bringmann, B., Gionis, A.: Mining Graph Evolution Rules. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part I. LNCS, vol. 5781, pp. 115–130. Springer, Heidelberg (2009)
19. Luigi, P., Pasquale, F., Carlo, S., Mario, V.: A subgraph isomorphism algorithm for matching large graphs. IEEE Trans. Pat. Anal. Mach. Intell. 26(10), 1367–1372 (2004)
20. Barabási, A., Albert, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)
21. Fagin, R.: Probabilities on finite models. J. of Symbolic Logic 41(1), 50–58 (1976)
22. Kibriya, A., Ramon, J.: Nearly exact mining of frequent trees in large networks. In: Proceedings of ECML-PKDD 2012 (in press)

Efficient Graph Kernels by Randomization

Marion Neumann[1], Novi Patricia[1], Roman Garnett[2], and Kristian Kersting[1]

[1] Knowledge Discovery Department, Fraunhofer IAIS,
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
{Marion.Neumann,Novi.Patricia,Kristian.Kersting}@iais.fraunhofer.de
[2] Robotics Institute, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213, United States
rgarnett@cs.cmu.edu

Abstract. Learning from complex data is becoming increasingly important, and graph kernels have recently evolved into a rapidly developing branch of learning on structured data. However, previously proposed kernels rely on having discrete node label information. In this paper, we explore the power of continuous node-level features for propagation-based graph kernels. Specifically, propagation kernels exploit node label distributions from propagation schemes like label propagation, which naturally enables the construction of graph kernels for partially labeled graphs. In order to efficiently extract graph features from continuous node label distributions, and in general from continuous vector-valued node attributes, we utilize randomized techniques, which easily allow for deriving similarity measures based on propagated information. We show that propagation kernels utilizing locality-sensitive hashing reduce the runtime of existing graph kernels by several orders of magnitude. We evaluate the performance of various propagation kernels on real-world bioinformatics and image benchmark datasets.

1 Introduction

For attribute-valued data, sophisticated kernel approaches for classification and regression have been widely and successfully studied. Nowadays, however, the bulk of information, such as available on the world wide web, is complex and highly structured. Structured data is commonly represented by graphs, which capture relations among entities, but also naturally model the structure of whole objects. Real-world examples are proteins or molecules in bioinformatics, image scenes in computer vision, text documents in natural language processing, and object and scene models in robotics, to name but a few. Learning in such domains and in turn developing meaningful kernels to take the structure of these data into account is becoming more and more important.

In addition to the structural properties of data entities, we often have access to vast quantities of additional, possibly continuous related information, for instance meta-data for images or text documents. Incorporating such information consistently is difficult and incomplete data and missing information constitute major challenges for learning. Unfortunately, existing graph kernels [4,6,10,18,19]

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 378–393, 2012.

rely on having discrete node labels and, besides, can only handle graphs with full node label information in a principled manner. In this paper, we propose the family of *propagation kernels* which leverage the power of continuous label distributions.

Triggered by previously introduced kernels on probabilistic models [8, 21], propagation kernels exploit distributions from propagation schemes like label propagation (LP) and enhance existing graph kernel frameworks to handle continuous, vector-valued node attributes. In particular, we define kernel inputs, i.e. graph features, that are counts of similar node label distributions on the respective graphs. This generalization additionally enables us to define graph kernels for partially labeled graphs in a natural way. Unfortunately, comparing all distributions of node labels among all graphs in the database scales as $\mathcal{O}(n^2)$, where n is the total number of nodes, and aggregating all distributions on node labels in one graph to a single vector leads to significant information loss. Hence, in order to efficiently determine the similarity among node label distributions and in general to be able to deal with continuous, vector-valued node attributes, we leverage randomization techniques from the theoretical computer science community. We define locality-sensitive hash (LSH) functions to create distance-preserving signatures for each node label distribution. These functions provide a randomized algorithm that allow us to efficiently compute the distribution-based count features for our kernels in $\mathcal{O}(n)$. We are able to show that the hash values can preserve both the total variation and the Hellinger metrics. Therefore, our LSH enables us to efficiently retrieve similar distributions based on these distance measures for probability distributions.

To summarize, the main contribution of our work is the introduction of a family of fast graph kernels based on propagating node labels. Specifically, we introduce locality-sensitive hash functions to efficiently compute the count features based on the similarity of node label distributions. Furthermore, we show that applying randomized techniques reduces the running time of existing graph kernels, namely kernels based on the Weisfeiler–Lehman test of isomorphism [18], by several orders of magnitude and we propose propagation kernels utilizing locality-sensitive hashing that are even more efficient.

We proceed as follows. We start off by touching upon related work. After introducing the family of propagation kernels and giving several examples thereof, we will describe locality-sensitive hashing for handling vector-valued node label distributions. Before concluding, we present experimental results for graph classification tasks on state-of-the-art image datasets, and commonly used bioinformatics benchmark datasets.

2 Related Work

Propagation kernels are related to three lines of research. First of all, they are deeply connected to several graph kernels developed within the graph mining community. Graph kernels can be categorized mainly into four classes: graph kernels based on walks [4, 10, 22] and paths [2], graph kernels based on limited-size subgraphs [7, 19], graph kernels based on subtree patterns [13, 16], and graph

kernels based on structure propagation [18]. Whereas the efficient kernel computation such as [22] are able to compare unlabeled graphs efficiently, Shervashidze *et al.* [19] specifically consider efficient comparisons of large, labeled graphs. The Weisfeiler–Lehman (WL) subtree kernel, one instance of the family of WL-kernels introduced, essentially computes count features for each graph based on the signatures arising from iterative multi-set label determination and compression steps. In every kernel iteration, these features are then the inputs to a base kernel and the WL-kernel is the sum of those base kernels over the iterations. The challenge of comparing large, partially labeled graphs—as considered by the propagation kernels introduced in the present paper—remains to a large extent unsolved. One could mark unlabeled nodes with a unique symbol and propagate this symbol using the Weisfeiler–Lehman kernels [18]. However, this neglects any label proportion information due to the diffusion process of labels on the graph. Likewise, one could just propagate labels across the graph and then run the WL-kernel. This, however, is also likely to neglect label proportion information. Indeed, after label propagation converges, we may ignore the label proportions of a node. Before convergence, however, we shall be concerned with information encoded in intermediate label proportions at nodes. Moreover, a two-stage approach may run many unnecessary label propagation iterations.

Second, propagation kernels are deeply connected to several recent lifted message-passing approaches [1,11,15,20] to probabilistic inference. They have rendered many of these large, previously intractable problems quickly solvable by exploiting the induced redundancies. Specifically, they automatically group nodes and potentials of the graphical model into supernodes and superpotentials if they have identical computation trees (i.e., the tree-structured "unrolling" of the graphical model computations rooted at the nodes). Then, they run modified message-passing approaches on this lifted (compressed) network. It is actually easy to see that the color-passing approach in [11] for computing the lifted network is as a form of the 1-dimensional Weisfeiler–Lehman algorithm as also employed by the WL-kernels [18]. However, there is a subtle difference. For lifted inference, symmetries among random variables easily break when variables become correlated by virtue of sharing asymmetrically observed evidence, that is labels of nodes are observed. Consequently, color-passing provides a lifted model that is not far from propositionalized, therefore canceling the benefits of lifted inference. For graph kernels, this is exactly what we want. The correlations help us to distinguish different graphs. This insight was the seed that grew into the idea of propagation kernels.

Finally, propagation kernels make another contact point, namely between WL-kernels and kernels that accommodate probability distributions [8,9,12,14]. However, whereas the latter ones essentially build kernels based on the outcome of probabilistic inference after convergence, propagation kernels intuitively count common sub-distributions induced after each iteration of running inference in two graphs. In doing so, they are able to take structure information into account.

3 Propagation Kernels

In the following, we introduce the general family of *propagation kernels* and present several instances based on propagating label information. The main insight here is, that the intermediate node label distributions, e.g., the iterative distribution updates of a label propagation scheme, capture label *and* structure information of the graphs. Hence, we design kernel inputs, i.e. graph features, based on the counts of similar distributions among the respective graphs' nodes. Further, we show that propagation kernels generalize existing structure propagating WL-kernels, especially the WL-subtree kernel [18].

3.1 General Definition

Here we will define a similarity measure $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ among graph instances $G^{(i)} \in \mathcal{X}$, in particular, let K be a positive semidefinite covariance function. Let $G_t^{(i)} = (V^{(i)}, E^{(i)}, L_t^{(i)})$ with $t = [0, .., T]$ be a sequence of graphs in $\mathcal{X}$, where $V^{(i)}$ is the set of nodes and $E^{(i)}$ is the set of edges. Further, each node in $V^{(i)}$ is endowed with one of k true labels and each graph has n_i nodes. $L_t^{(i)} \in \mathbb{R}^{n_i \times k}$ represents the label distributions[1] for all nodes in $V^{(i)}$ which are iteratively updated. Note that we do not assume that the node labels have to be given for all nodes. Propagation kernels can naturally be computed for partially labeled graphs as the features are only built upon the node label distributions, which can be initialized uniformly for unknown node labels. Observed node labels are represented by a trivial delta distribution.

Propagation kernels are defined by applying the following iterative procedure $T + 1$ times, beginning with an initial set of graphs $\{G_0^{(i)}\}$ with label distributions initialized as above.

Step 1: count common node label distributions. First, we generate feature vectors $\phi(G_t^{(i)})$ for each graph by counting common label distributions induced over the nodes among the respective graphs. Therefore, each node in each graph is placed into one of a number of "bins," each one collecting similar label distributions, and these vectors count the nodes in each bin for each graph. The exact details of this procedure are given below, in Section 3.2 and Section 4.
Step 2: calculate current kernel contribution. Given these vectors, for each pair of graphs $G^{(i)}$ and $G^{(j)}$, we calculate

$$k(G_t^{(i)}, G_t^{(j)}) = \langle \phi(G_t^{(i)}), \phi(G_t^{(j)}) \rangle, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is an arbitrary base kernel. This value will be an additive contribution to the final kernel value between these graphs.
Step 3: propagate node label distributions. Finally, we apply an iterative update scheme for the node label distributions

$$L_t^{(i)} \to L_{t+1}^{(i)}, \tag{2}$$

[1] Note that $L^{(i)}$ could also involve continuous, vector-valued node attributes, however, in this paper we focus on label distributions.

Algorithm 1. The general propagation kernel computation.

given iterations T, initial label distributions $L_0^{(i)}$, base kernel $k(\cdot,\cdot)$
$K \leftarrow 0$
for $t \leftarrow 0 \dots T$ **do**
  for all graphs $G^{(i)}$ **do**
    $\phi(G_t^{(i)}) \leftarrow 0$
    for $j \leftarrow 1 \dots n_i$ **do**
      $\phi(G_t^{(i)}) \leftarrow \phi(G_t^{(i)}) + f(\ell_{t,j}^{(i)})$ ▷ count node label distributions, Eq. (4)
    end for
    $L_t^{(i)} \rightarrow L_{t+1}^{(i)}$ ▷ update label distribution, Eq. (2)
  end for
  $K \leftarrow K + k(\Phi, \Phi)$ ▷ Φ is $N \times k''$ matrix of ϕ vectors
end for

e.g. label propagation. These new label distributions replace those in the current set of graphs, and we continue with Step 1. The exact choice for this update results in different propagation kernels; examples are provided in Section 3.3.

Finally, the T-iteration propagation kernel between two graphs $G^{(i)}$ and $G^{(j)}$ is defined as

$$K_T(G^{(i)}, G^{(j)}) = \sum_{t=0}^{T} k(G_t^{(i)}, G_t^{(j)}). \tag{3}$$

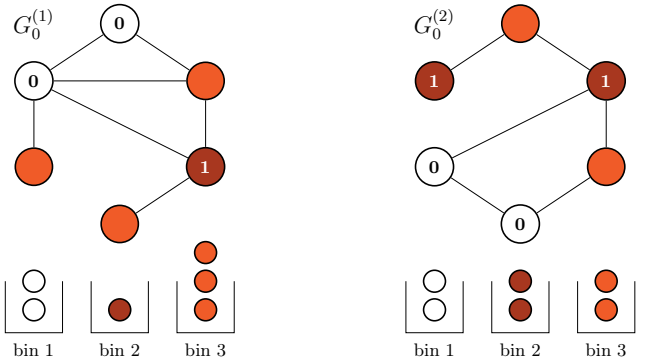
The propagation kernel computation is summarized in Algorithm 1 and an illustrative example for $t = 0$ and $t = 1$ for two graphs is shown in Figure 1.

3.2 Distribution-Based Graph Features

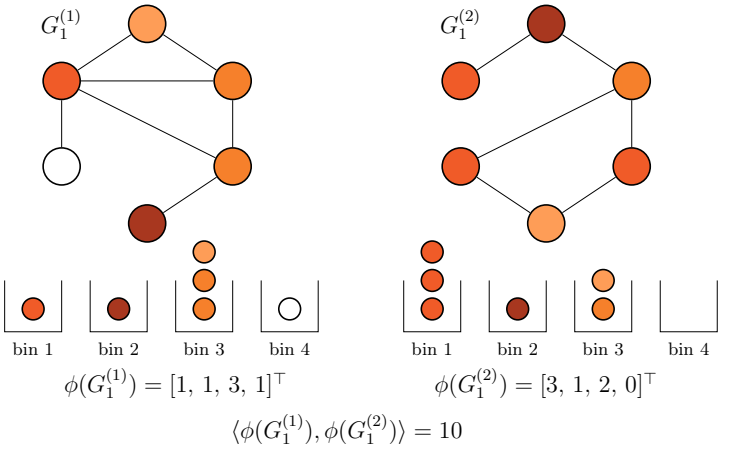
The main ingredient of propagation kernels is the way distribution-based graph features are generated. Let $\ell_{t,j}^{(i)}$ be the j-th row of $L_t^{(i)}$ and $\mathcal{L} = \bigcup_i^N \bigcup_j^{n_i} \{\ell_{t,j}^{(i)}\}$ be the set of all uniquely occurring label distributions on the nodes of all graphs. The family of propagation kernels is characterized by generating graph features by counting node-level features on that graph. This will be captured by a function f mapping from the space of distributions $\mathbb{R}^k$ into the space of standard basis vectors $E_{k'} = \{e_1, ..., e_{k'}\}$ with $k' = |\mathcal{L}| \leq n$, where n is the number of nodes for all graphs $n = \sum_{i=1}^{N} n_i$. We now define

$$\phi(G_t^{(i)}) = \sum_{j=1}^{n_i} f(\ell_{t,j}^{(i)}). \tag{4}$$

As the node label distributions $\ell_{t,j}^{(i)}$ are $k-$dimensional *continuous* vectors the cardinality of $\mathcal{L}$ might in fact be equal to the total number of nodes n in the whole graph database. This, however, means that our derived features are not meaningful as, in this case, we never get the same count feature for any two similarly distributed nodes and the kernel value for any two graphs as defined



(a) Initial label distributions and base kernel value for $t = 0$



(b) Updated label distributions and base kernel value for $t = 1$

Fig. 1. Propagation Kernel Propagation kernel computations for two graphs $G_t^{(1)}$ and $G_t^{(2)}$ with binary node labels using one iteration of label propagation, Eq.(6), as distribution update. Node label distributions are decoded by color, white means $\ell_{0,j}^{(i)} = [1, 0]$ and dark red stands for $\ell_{0,j}^{(i)} = [0, 1]$, the initial distributions for unlabeled nodes (light red) are $\ell_{0,j}^{(i)} = [1/2, 1/2]$. Panel (a) shows the initial distributions, bins, and respective kernel computation and panel (b) depicts distributions, bins, features and linear base kernel for $t = 1$.

in Eq. (1) is always zero. To ensure the acquisition of meaningful features we leverage *quantization* [5]. Hence, the mapping f is replaced by $q \colon \mathbb{R}^k \to E_{k'}$, where q is a quantization function such that $k' \ll |\mathcal{L}| \leq n$. Note, that deriving a quantization function for distributions involves considering distance metrics

for distributions such as Hellinger or total variation (TV) distance. We are not defining the quantization function here but instead give an efficient solution by locality-sensitive hashing [3] in Section 4 and also show that we can construct quantization functions which are Hellinger and TV distance preserving.

3.3 Instances of Propagation Kernels

So far, we defined the general family of propagation kernels. Specific choices of label update schemes, cf. Eq. (2), result in different instances of the propagation kernel family. In particular, we introduce the *diffusion graph kernel*, the *label propagation kernel*, the *belief propagation kernel*, and *structure propagation kernels* as for instance the WL-subtree kernel.

Diffusion Graph Kernel: For the diffusion graph kernels we use the following update for the node label distributions $L_t^{(i)} \rightarrow L_{t+1}^{(i)}$. Given the adjacency matrix $A^{(i)}$ of graph $G^{(i)}$ label diffusion on each node is defined as

$$L_{t+1}^{(i)} \leftarrow T^{(i)} L_t^{(i)}, \tag{5}$$

where $T^{(i)}$ is the transition matrix, i.e., the row-normalized adjacency matrix $T^{(i)} = (D^{(i)})^{-1} A^{(i)}$, where $D^{(i)}$ is the diagonal degree matrix with $D_{aa}^{(i)} = \sum_b A_{ab}^{(i)}$.

Label Propagation Kernel: The label distribution update for the label propagation kernel differs in the fact, that before each iteration of label diffusion the labels of the originally labeled nodes are *pushed back* [25]. Let $L_0^{(i)} = \left[L_{0,[labeled]}^{(i)}, L_{0,[unlabeled]}^{(i)}\right]^\top$ be the original labels of graph $G^{(i)}$, where the distributions in $L_{0,[labeled]}^{(i)}$ represent hard labels and $L_{0,[unlabeled]}^{(i)}$ are initialized by a uniform label distribution, i.e., each entry is $1/k$. Then the label propagation is defined by

$$\begin{aligned} L_{t,[labeled]}^{(i)} &\leftarrow L_{0,[labeled]}^{(i)}, \\ L_{t+1}^{(i)} &\leftarrow T^{(i)} L_t^{(i)}. \end{aligned} \tag{6}$$

Note, that we can choose other step sizes for the label propagation update scheme as one. This means that we run several iterations of label propagation for each distribution update. This can be beneficial for settings with partially labeled graphs. Further, other update schemes, such as "label spreading" [24], can be used in a similar manner resulting in a *label spreading kernel*.

Belief Propagation Kernel: Triggered by the idea of defining the similarity of graphical models, like conditional random field (CRF) representations of images or different groundings of a Markov logic network (MLN) [17], we can also use belief propagation [23] to get the node-level feature update in Eq. (2). To define the belief propagation kernel, we simply use marginal probabilities instead of

label distributions. Due to space limitations, we do not provide any more details here and leave comprehensive derivations and experiments for future work.

Structure Propagation Kernel — The WL-subtree Kernel: The WL graph kernels for labeled graphs are currently state-of-the-art considering both prediction performance and runtime [18]. Therefore, we briefly introduce one instance, the WL-subtree kernel, and give its definition within our proposed framework of propagation kernels. Given two graphs $G^{(i)}$ and $G^{(j)}$, the subtree pattern kernel counts all pairs of matching substructures in subtrees rooted at all nodes of $G^{(i)}$ and $G^{(j)}$ respectively. The runtime complexity of this approach for N graphs is $\mathcal{O}(n^2T4^d)$ [16], where d is the maximum node degree in all graphs. The idea of using the 1-dimensional Weisfeiler–Lehman test of isomorphism is to overcome the poor ability to scale to large, labeled graphs, as it scales linearly on the number of edges of the graphs and the length of the considered graph sequence. Hence, the WL-subtree kernel on N graphs with T iterations can be computed in $\mathcal{O}(Tm + NTn)$ [18], where m is the total number of edges in all graphs and n is the total number of nodes. Given two graphs $G^{(i)}$ and $G^{(j)}$, the algorithm works as follows. First, a signature is generated for each node in each graph by concatenating its label with a sorted multiset of its neighboring nodes. Then each node is assigned a new label such that nodes with the same signature are labeled the same. This means a hard label update in Eq. (2). Indeed, the WL-subtree kernel can be defined analogously to Eq. (3). The sequence of graphs $\{G_t^{(i)}\}$ is given by hard labels reflecting the signatures for the respective subtree pattern and the features are represented by

$$\phi_{\text{WL}}(G_t^{(i)}) = \sum_{j=1}^{n_i} g(\ell_{t,j}^{(i)}), \tag{7}$$

where $\ell_{t,j}^{(i)} \in \{e_1, ..., e_{k''}\}$ with $k'' \leq n$ being the number of different subtree patterns and $g\colon i \mapsto e_i$. That means, $\ell_{t,j}^{(i)}$ represents a hard signature for every node j and g maps each signature to one standard basis vector. Indeed, this view opens up the possibility to handle probabilistic subtree patterns in the setting of WL-kernels for partially labeled graphs, which, in turn, is a compelling direction to enhance the power of WL-kernels.

4 Locality-Sensitive Hashing for Propagation Kernels

We now describe our quantization approach for implementing propagation kernels on graphs with node label distributions. We take our inspiration from locality-sensitive hashing [3], which seeks for quantization functions on metric spaces where points "close enough" to each other in that space are "probably" assigned to the same bin. In our case, we will consider each node label vector as being an element of the space of discrete probability distributions on k items equipped with an appropriate probability metric.

We will begin with a formal definition. Let $\mathcal{X}$ be a metric space with metric $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and let $\mathcal{Y} = \{1, 2, \ldots, k'\}$. Let $\theta > 0$ be a threshold, $c > 1$ be

an approximation factor, and $p_1, p_2 \in (0,1)$ be the given success probabilities. A set of functions $\mathcal{H}$ from $\mathcal{X}$ to $\mathcal{Y}$ is called a $(\theta, c\theta, p_1, p_2)$-*locality sensitive hash* (LSH) if for any function $h \in \mathcal{H}$ chosen uniformly at random, and for any two points $x, x' \in \mathcal{X}$, we have that

- if $d(x,x') < \theta$, then $\Pr(h(x) = h(x')) > p_1$, and
- if $d(x,x') > c\theta$, then $\Pr(h(x) = h(x')) < p_2$.

It is known that we can construct LSH families for L^p spaces with $p \in (0,2]$ [3]. Let V be a real-valued random variable. V is called p-stable if for any $\{x_1, x_2, \ldots, x_d\}$, $x_i \in \mathbb{R}$ and independently sampled $v_1, v_2, \ldots, v_d$, we have $\sum x_i v_i \sim \|\mathbf{x}\|_p V$. Explicit p-stable distributions are known for some p; for example, the standard Cauchy distribution is 1-stable, and the standard normal distribution is 2-stable. Given the ability to sample from a p-stable distribution V, we may define a LSH $\mathcal{H}$ on $\mathbb{R}^d$ with the L^p metric [3]. An element h of $\mathcal{H}$ is specified by three parameters: a width $w \in \mathbb{R}^+$, a d-dimensional vector $\mathbf{v}$ whose entries are independent samples of V, and $b \in [0,w]$ drawn from $\mathcal{U}[0,w]$, and defined as

$$h(\mathbf{x}; w, \mathbf{v}, b) = \left\lfloor \frac{\mathbf{v}^\top \mathbf{x} + b}{w} \right\rfloor . \tag{8}$$

We may now consider $h(\cdot)$ to be a function mapping our label distributions to integer-valued bins, where similar distributions end up in the same bin. If we number the non-empty integer bins occupied by all the nodes in all graphs from 1 to k'', then we may define the function f in Eq. (4) by $f(\cdot) = u \circ h(\cdot)$, where $u : \mathbb{N} \to E_{k''}$. To decrease the probability of collision it is common to choose more than one random vector $\mathbf{v}$. For propagation kernels, however, we only use one hyperplane, as we effectively have T hyperplanes for the whole kernel computation and the probability of a hash conflict is reduced over the iterations.

The intuition behind the expression in Eq. (8) is that p-stability implies that two vectors that are close under the L^p norm will be close after taking the dot product with $\mathbf{v}$; specifically, $(\mathbf{v}^\top \mathbf{x} - \mathbf{v}^\top \mathbf{x}')$ is distributed as $\|\mathbf{x} - \mathbf{x}'\|_p V$. In our applications, we are concerned with the space of discrete probability distributions on k elements, endowed with a probability metric d. Here we specifically consider the *total variation* (TV) and *Hellinger* (H) distances:

$$d_{\mathrm{TV}}(p,q) = 1/2 \sum_i |p_i - q_i|, \quad d_{\mathrm{H}}(p,q) = \left(1/2 \sum_i \left(\sqrt{p_i} - \sqrt{q_i} \right)^2 \right)^{1/2} .$$

The total variation distance is simply half the L^1 metric, and the Hellinger distance is a scaled version of the L^2 metric after applying the map $p \mapsto \sqrt{p}$. We may therefore create a locality-sensitive hash family for d_{TV} by direct application of Eq. (8), and create a locality-sensitive hash family for d_{H} by applying Eq. (8) after applying the square root map to our label distributions. These are the quantization schemes applied in our experiments.

5 Empirical Evaluation

Our intention here is to investigate the power of propagation kernels for graph classification. Specifically, we investigate the two following questions:

(Q1) Do propagation kernels utilizing continuous distribution-based features arising from propagating label information perform competitively as state-of-the-art graph kernels for graph classification?
(Q2) Does randomization, in particular locality-sensitive hashing techniques, improve efficiency over state-of-the-art graph kernels?
To this aim, we implemented propagation kernels in Matlab and considered the following experimental protocol.

5.1 Experimental Protocol

We compare classification accuracy and runtime for several different instances of propagation kernels: the *diffusion graph kernel* and the *label propagation kernel* with Hellinger distance and total variation distance, and a *structure propagation kernel*, namely the WL-subtree kernel. We choose the WL-subtree kernel for comparisons as it is currently the most accurate and efficient graph kernel and additionally report several results for other graph kernels from [18].

We consider two general settings, graph classification for fully and for partially labeled graphs. The latter is a reasonable situation when dealing with semantic images as fully labeled data is costly or even impossible to acquire. Note, that for partially labeled datasets we use the *label propagation kernel*, whereas for fully labeled graphs this does not make sense because of the *push back* of original labels. Here, we choose the *diffusion graph kernel*.

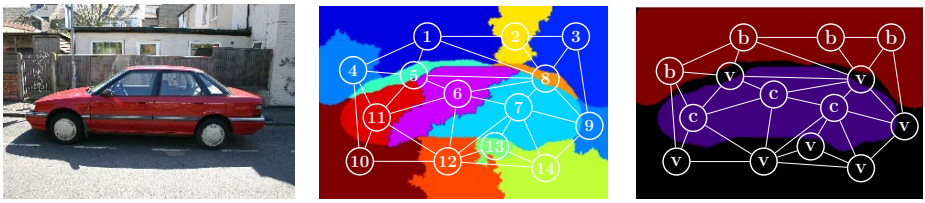
The classification performance is evaluated by running C-SVM classifications using libSVM,[2] where the cost parameter is learned by cross-validation on the training set. We compute all kernels for $T = 0, ..., 10$ and report the average of the best accuracies from 10 re-runs of a 10-fold cross-validation. For all runtime experiments all kernels are as well computed for $T = 10$ and all experiments were conducted on an Apple Mac Pro workstation with two 2.26 GHz quad-core Intel Xeon "Gainestown" processors (model E5520) and 28 GB of RAM. We used a linear base kernel for all methods and the bin width parameter for LSH was set to $w = 10^{-5}$. All results were fairly insensitive to the exact choice of w.

5.2 Datasets

We considered two real-world benchmark datasets.

Bioinformatics Benchmark Data: We ran experiments on the following benchmark datasets: MUTAG, ENZYMES, NCI1, NCI109, and D&D. MUTAG contains 188 sets of mutagenic aromatic and heteroaromatic nitro compounds, the label refers to their mutagenic effect on the Gram-negative bacterium *Salmonella*

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm/



(a) Original image (b) Graph over superpixels (c) Semantic labels

Fig. 2. Graph Based Scene Classification In semantic scene classification the original images (a) are represented by graphs of superpixels (b). Each superpixel node has an attached semantic label (c) (here: b = *building*, c = *car*, and v = *void*). Each graph represents the semantic scene of an image and similar scenes are classified according to a kernel capturing label and structure information.

typhimurium. ENZYMES has 6 EC top-level classes. It is a dataset of protein tertiary structures belonging to 600 enzymes from the BRENDA enzyme database. NCI1 and NCI109 are anti-cancer screens, in particular for cell lung cancer and ovarian cancer cell lines, respectively. D&D consists of 1178 protein structures, with the nodes in each graph represent amino acids and two nodes forming an edge if they are less than 6 Ångstroms separated. For a more comprehensive introduction and references, see [18].

Image Benchmark Data: The two real-world image datasets MSRC 9-class and MSRC 21-class[3] are state-of-the-art datasets in semantic image processing. Each image is represented by a conditional Markov random field graph, as illustrated in Figure 2. The nodes of each graph are derived by oversegmenting the images using the quick shift algorithm[4] with an average of 40 superpixels per graph. Hence, each node represents one superpixel and the semantic (ground-truth) node labels are derived by taking the mode ground-truth label of all pixels in the corresponding segment. Note, that the number of nodes varies from graph to graph. MSRC9 consists of 221 images, and a total of 8969 nodes. The node labels consist of nine classes *building, grass, tree, cow, sky, aeroplane, face, car, bicycle* and a label *void* to handle objects that do not fall into one of these classes. Each image can be classified into one out of eight classes. From the MSRC 21-class dataset, which is a more comprehensive and complex image dataset, we derived two datasets for our experiments. MSRC21 consists of 565 images with 24 109 labeled superpixels of 21 classes: *building, grass, tree, cow, sheep, sky, airplane, water, face, car, bicycle, flower, sign, bird, book, chair, road, cat, dog, body, boat,* and *void*. For the second dataset, MSRC21C, we extracted a subset of the most challenging scenes by removing all images having fewer than four different class labels. The resulting dataset consists of 209 graphs and 8 626 nodes in total. For both datasets based on the MSRC 21-class data each image can be classified as one out of 20 classes.

[3] http://research.microsoft.com/en-us/projects/ObjectClassRecognition/
[4] http://www.vlfeat.org/overview/quickshift.html

Table 1. Average accuracy (and standard deviation) on the image datasets for *diffusion graph kernel* K_{DIFF} using Hellinger distance (+H) resp. total variation distance (+TV), and for the *WL-subtree kernel* K_{WL}. Bold indicates best result.

	dataset		
method	MSRC9	MSRC21	MSRC21C
$K_{\text{DIFF+H}}$	91.6 (0.5)	**83.6** (0.8)	**88.7** (0.7)
$K_{\text{DIFF+TV}}$	91.6 (0.5)	**83.6** (0.8)	**88.7** (0.7)
K_{WL}	**92.1** (0.8)	82.2 (1.1)	88.5 (0.4)

Table 2. Average accuracy (and standard deviation) on the bioinformatics benchmark datasets for *diffusion graph kernel* K_{DIFF} using Hellinger distance (+H) resp. total variation distance (+TV), and for the *WL-subtree kernel* K_{WL}. Bold indicates best result. Results for random walk kernel and Ramon–Gärtner kernel are taken from [18] to provide a broader overview of state-of-the-art graph kernel performances; however, please note that we did not re-run the experiments and hence they have most likely been produced using different random folds.

	dataset				
method	MUTAG	ENZYMES	NCI1	NCI109	D&D
$K_{\text{DIFF+H}}$	**87.7** (1.3)	47.1 (1.2)	**84.4** (0.2)	**84.0** (0.3)	79.2 (0.4)
$K_{\text{DIFF+TV}}$	87.5 (1.3)	47.0 (1.1)	84.2 (0.3)	83.6 (0.3)	79.3 (0.3)
K_{WL}	87.0 (1.0)	**53.1** (1.3)	82.2 (0.2)	82.5 (0.2)	**80.0** (0.4)
random walk [22]	80.7 (0.4)	21.7 (0.9)	64.3 (0.3)	63.5 (0.2)	71.7 (0.5)
Ramon–Gärtner [16]	85.7 (0.5)	13.4 (0.9)	61.9 (0.3)	61.7 (0.2)	57.2 (0.1)

Table 3. Average accuracy (and standard deviation) on 10 different sets of partially labeled images for *label propagation kernel* using TV distance ($K_{\text{LP+TV}}$), and for the *WL-subtree kernel* with unlabeled nodes treated as additional label K_{WL} and with hard labels derived from converged LP (LP + K_{WL}).

		labels missing			
dataset	method	20%	40%	60%	80%
MSRC9	$K_{\text{LP+TV}}$	**90.0** (1.2)	**88.7** (1.0)	**86.6** (1.3)	**80.4** (1.8)
	LP + K_{WL}	**90.0** (0.6)	87.9 (1.9)	83.2 (2.0)	77.9 (3.1)
	K_{WL}	89.2 (1.5)	88.1 (1.5)	85.7 (1.9)	78.5 (2.7)
MSRC21	$K_{\text{LP+TV}}$	**86.9** (0.8)	**84.7** (1.0)	**79.5** (0.9)	**69.3** (1.1)
	LP + K_{WL}	85.8 (0.6)	81.5 (0.8)	74.5 (1.0)	64.0 (1.2)
	K_{WL}	85.4 (1.3)	81.9 (1.2)	76.0 (0.8)	63.7 (1.3)

5.3 Experimental Results

Under our experimental protocol, propagation kernels gave the following results.

(Q1) Predictive Performance: The predictive performances for fully labeled graphs are summarized in Tables 1 and 2. On MSRC21, MSRC21C, MUTAG, NCI1, and NCI109, propagation kernels reached the highest accuracy. Only on ENZYMES, did K_{WL} perform considerably better than propagation kernels. For the remaining datasets, the predictive performance is comparable. The Ramon–Gärtner and random-walk kernels were less competitive to the propagation kernels. To assess the predictive performance of propagation kernels on partially labeled graphs, we ran the following experiments 10 times. We randomly removed 20–80% of the labels in MSRC9, MSRC21, and MSRC21C and computed cross-validation accuracies and standard deviations. Because the WL-subtree kernel was not designed for partially labeled graphs, we compare the *label propagation kernel* to two variants: one where we treat unlabeled nodes as an additional label "*u*" (K_{WL}) and another where we use hard labels derived from running label propagation until convergence ($\text{LP} + K_{\text{WL}}$). The results for the first two datasets are shown in Table 3. The results for MSRC21C showed the same behavior and hence were omitted. For larger fractions of missing labels $K_{\text{LP+TV}}$ obviously outperforms the baseline methods and surprisingly running label propagation until convergence and then the WL-subtree kernel gives poorer results than K_{WL}. However, label propagation might be beneficial for larger amounts of missing labels. In general, the results on all experiments clearly show that question **(Q1)** can be answered affirmatively.

(Q2) Running Time: The runtime results are summarized in Table 4. Empirically, we observe that propagation kernels can be orders of magnitude faster then existing graph kernels. They can easily scale to graphs with thousands of nodes. On D&D, $K_{\text{DIFF+TV}}$ was computed at least twice as fast as any other method. On ENZYMES, $K_{\text{DIFF+TV}}$ takes less then a second, whereas all other methods take several seconds. Compared to $K_{\text{WL(REF)}}$, it is two orders of magnitude faster. Comparing the runtimes of $K_{\text{WL(LSH)}}$ and $K_{\text{WL(REF)}}$, we clearly see that leveraging randomization significantly outperforms the non-randomized approach. We also compared the runtime of propagation kernels using label propagation to the WL-subtree kernel on the MSRC21 dataset with partially labeled graphs. We again compare $K_{\text{LP+TV}}$ with $K_{\text{WL(LSH)}}$ and $\text{LP} + K_{\text{WL(LSH)}}$. The results are summarized in Figure 3. $K_{\text{WL(REF)}}$ is over 36 times slower than $K_{\text{LP+TV}}$. These results again confirm that propagation kernels have attractive scalability properties for large datasets. The $\text{LP} + K_{\text{WL}}$ approach wastes computation time while running LP to convergence before it can even begin calculating the kernel. The intermediate label distributions obtained during the convergence process are already extremely powerful for classification and allow one to save computation time. These results clearly answer question **(Q2)** affirmatively.

To summarize, propagation kernels turned out to be competitive in terms of predictive accuracy and speed on all datasets, often by orders of magnitude. Thus, questions **(Q1)** and **(Q2)** can be answered affirmatively.

Table 4. Runtime in seconds for $T = 10$ on the bioinformatics datasets for the *diffusion graph kernel* ($K_{\text{DIFF+TV}}$) using TV distance, and for the *WL-subtree kernel* for a implementation leveraging randomization ($K_{\text{WL(LSH)}}$) and the standard implementation ($K_{\text{WL(REF)}}$) presented in [18]. Bold indicates best result. Results for WL-edge kernel ($K_{\text{WL-EDGE}}$) and graphlet count kernel are taken from [18] to provide a broader overview of state-of-the-art graph kernel performances.

	dataset					
method	MUTAG	ENZYMES	NCI1	NCI109	D&D	total
$K_{\text{DIFF+TV}}$	0.12 s	**0.89 s**	116 s	133 s	**55 s**	**376 s**
$K_{\text{WL(LSH)}}$	**0.03 s**	1.6 s	185 s	189 s	117 s	493 s
$K_{\text{WL(REF)}}$	4.7 s	29 s	216 s	216 s	511 s	977 s
$K_{\text{WL-EDGE}}$ [18]	3 s	11 s	**65 s**	**58 s**	3 days	3 days
graphlet count [19]	3 s	5 s	87 s	87 s	23 hours	23 hours

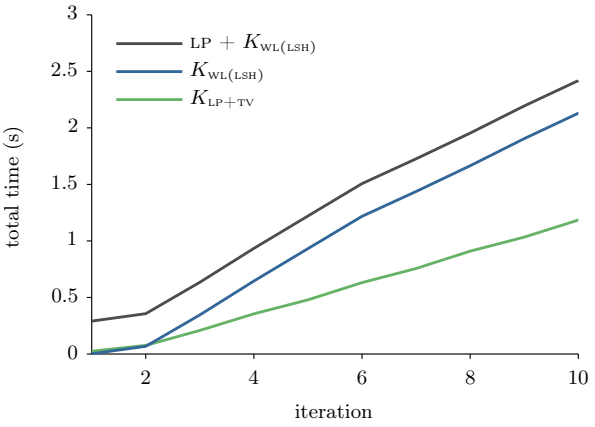


Fig. 3. Runtime for Partially Labeled MSRC21 Average time in seconds over 10 different instances of the MSRC21 dataset with 50% labeled nodes for kernel iterations T from 0 to 10. We compare the *WL-subtree kernel* with unlabeled nodes treated as additional label ($K_{\text{WL(LSH)}}$), and with hard labels derived from converged LP distributions (LP + $K_{\text{WL(LSH)}}$), and the *label propagation kernel* with TV distance ($K_{\text{LP+TV}}$). $K_{\text{WL(REF)}}$ required 36s for $T = 10$ and is not included.

6 Conclusions and Future Work

Probabilistic models provide a principled way of spreading information and even treating missing information within graphs. Known labels can be used to propagate information through the graph in order to label all nodes. On the other hand, discriminative methods such as support vector machines enable us to construct flexible decision boundaries and often result in classification performance

superior to that of the model based approaches. In this paper, we developed a natural way of combining both frameworks for the construction of graph kernels, called propagation kernels. Intuitively, propagation kernels count common sub-distributions induced in each iteration of running inference in two graphs. For counting the continuous information — the distributional information computed for each node — efficiently, they leverage the randomized technique of locality-sensitive hashing. As our experimental results demonstrate, propagation kernels are competitive in terms of accuracy with state-of-the-art kernels on several classification benchmark datasets, even reaching the highest accuracy level on five out of eight datasets. Moreover, in terms of runtime, propagation kernels outperform other graph kernels, even the recently developed efficient WL-kernels.

Propagation kernels provide several interesting avenues for future work. While we have used classification to guide the development of propagation kernels, the results are directly applicable to regression, clustering, and ranking, among other tasks. Employing message-based probabilistic inference schemes such as (loopy) belief propagation directly paves the way to deal with more general structures then just graphs; we are currently investigating Markov logic networks [17]. By considering the computation trees—the tree-structured unrolling of a given graph rooted at the nodes—one may even realize within-network relational classification using propagation kernels.

Acknowledgments. This work was partly supported by the Fraunhofer ATTRACT fellowship STREAM and by the European Commission under contract number FP7-248258-First-MM.

References

1. Ahmadi, B., Kersting, K., Sanner, S.: Multi-Evidence Lifted Message Passing, with Application to PageRank and the Kalman Filter. In: Proc. of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011 (2011)
2. Borgwardt, K.M., Kriegel, H.-P.: Shortest-path kernels on graphs. In: Proceedings of International Conference on Data Mining (ICDM 2005), pp. 74–81 (2005)
3. Datar, M., Indyk, P.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the 20th Annual Symposium on Computational Geometry (SCG 2004), pp. 253–262 (2004)
4. Gärtner, T., Flach, P.A., Wrobel, S.: On Graph Kernels: Hardness Results and Efficient Alternatives. In: Schölkopf, B., Warmuth, M.K. (eds.) COLT/Kernel 2003. LNCS (LNAI), vol. 2777, pp. 129–143. Springer, Heidelberg (2003)
5. Gersho, A., Gray, R.: Vector quantization and signal compression. Kluwer Academic Publishers, Norwell (1991)
6. Hido, S., Kashima, H.: A linear-time graph kernel. In: Proc. of the 9th IEEE International Conference on Data Mining (ICDM 2009), pp. 179–188 (2009)
7. Horváth, T., Gärtner, T., Wrobel, S.: Cyclic pattern kernels for predictive graph mining. In: Proceedings of Knowledge Discovery in Databases (KDD 2004), pp. 158–167 (2004)
8. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: Proc. of Neural Information Processing Systems (NIPS 1998), pp. 487–493 (1998)

9. Jebara, T., Kondor, R.I., Howard, A.: Probability product kernels. Journal of Machine Learning Research 5, 819–844 (2004)
10. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: Proc. of the 20th International Conference on Machine Learning (ICML 2003), pp. 321–328 (2003)
11. Kersting, K., Ahmadi, B., Natarajan, S.: Counting Belief Propagation. In: Proc. of the 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009 (2009)
12. Lafferty, J.D., Lebanon, G.: Information diffusion kernels. In: Proc. of Neural Information Processing Systems (NIPS 2002), pp. 375–382 (2002)
13. Mahé, P., Vert, J.-P.: Graph kernels based on tree patterns for molecules. Machine Learning 75(1), 3–35 (2009)
14. Moreno, P.J., Ho, P., Vasconcelos, N.: A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications. In: Proc. of Neural Information Processing Systems, NIPS 2003 (2003)
15. Neumann, M., Kersting, K., Ahmadi, B.: Markov logic sets: Towards lifted information retrieval using pagerank and label propagation. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence, AAAI 2011 (2011)
16. Ramon, J., Gärtner, T.: Expressivity versus efficiency of graph kernels. In: Proceedings of the 1st International Workshop on Mining Graphs, Trees and Sequences, pp. 65–74 (2003)
17. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
18. Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler–Lehman Graph Kernels. Journal of Machine Learning Research 12, 2539–2561 (2011)
19. Shervashidze, N., Vishwanathan, S.V.N., Petri, T., Mehlhorn, K., Borgwardt, K.M.: Efficient graphlet kernels for large graph comparison. Journal of Machine Learning Research - Proceedings Track 5, 488–495 (2009)
20. Singla, P., Domingos, P.: Lifted First-Order Belief Propagation. In: Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008), pp. 1094–1099 (2008)
21. Tsuda, K., Kawanabe, M., Rätsch, G., Sonnenburg, S., Müller, K.-R.: A new discriminative kernel from probabilistic models. Neural Computation 14(10), 2397–2414 (2002)
22. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R.I., Borgwardt, K.M.: Graph kernels. Journal of Machine Learning Research 11, 1201–1242 (2010)
23. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Generalized belief propagation. In: Proc. of Neural Information Processing Systems (NIPS 2000), pp. 689–695 (2000)
24. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Proc. of Neural Information Processing Systems, NIPS 2009 (2003)
25. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02-107. Carnegie Mellon University (2002)

Graph Mining for Object Tracking in Videos

Fabien Diot[1,2], Elisa Fromont[1], Baptiste Jeudy[1],
Emmanuel Marilly[2], and Olivier Martinot[2]

[1] Université de Lyon, Université Jean Monnet de Saint-Etienne
Laboratoire Hubert Curien, UMR CNRS 5516, 42000, Saint-Etienne, France
[2] Alcatel-Lucent Bell Labs, Centre de Villarceaux,
Route de Villejust, 91620, Nozay, France

Abstract. This paper shows a concrete example of the use of graph mining for tracking objects in videos with moving cameras and without any contextual information on the objects to track. To make the mining algorithm efficient, we benefit from a video representation based on dynamic (evolving through time) planar graphs. We then define a number of constraints to efficiently find our so-called spatio-temporal graph patterns. Those patterns are linked through an occurrences graph to allow us to tackle occlusion or graph features instability problems in the video. Experiments on synthetic and real videos show that our method is effective and allows us to find relevant patterns for our tracking application.

1 Introduction and Related Work

Object tracking in videos is a very popular research field in computer vision due to the numerous applications such as video-surveillance in very diverse environments (airports, cities, large public areas), pedestrian protection systems, automatic calibration methods using moving robots, tracking complicated surfaces, medical image applications etc. [11]. Most of the ongoing research [1,11] makes strong assumptions about the objects to track (people, car, etc.) which are often modelled in advance, or about the tracking context (stable background, object moving in a single direction, stable lighting conditions, etc.) to perform an efficient tracking. These methods rely on two steps, the object detection in the frame and the tracking process. For detection, techniques are based on frame difference or the use of background subtraction [12], optical flow (detection of the relative motion between a static camera and the filmed objects) [15] or background information on the objects to track (skin color, shape etc.). For the tracking process, techniques consist in predicting the next region (or contour) of interest using probabilistic or deterministic methods [7] (and then possibly add another detection step). They use some discriminant features attached to the objects and/or use apriori learned models of the objects which can possibly be updated during the tracking step [13].

In this work, we would like to show how data mining and in particular graph mining can help to track multiple objects in a video in the specific case in which both the objects and the background are moving and when no supervised

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 394–409, 2012.

information about the objects to track is known in advance (which could allow to learn some models apriori). We regard a video as a dynamic graph, whose evolution over time is represented by a series of *plane* graphs, one graph for each video frame. The graph representing each frame is a region adjacency graph (RAG) [6]. In RAGs, the barycenters of the different regions in a frame are the nodes of the graph, and an edge exists between two nodes if the regions are adjacent in the frame. By representing a video as a series of plane labelled graphs, subgraph patterns in this series may correspond to objects that frequently appear in a video, such as the planes in the frames of Fig. 2 and 3.

This paper is based on [14] where we have already assessed the interest of our plane graph mining algorithm called PLAGRAM compared to a generic graph mining algorithm such as GSPAN [16] on which it is based. PLAGRAM can efficiently mine a dynamic graph representing a video (i.e. a plane graphs database). Note that most existing algorithms which mine dynamic graphs (e.g., dynamic networks) consider graphs with only edges insertions or deletions i.e., the time series of graphs share the same set of nodes over time (see, e.g.,[3]), or in which nodes and edges are only added and never deleted (see, e.g., [2]). In [5,17], the problem is to mine spatio-temporal relationships between moving objects (the mined relationships are restricted to some predefined graphs like cliques, star graphs or sequences). In our approach, however, there is no information about the correspondence between the nodes in one graph (video frame) and those in the others. In [14], some simple constraints were used in a post processing step to obtain some so-called spatio-temporal patterns. However, the definition of spatio-temporal patterns (and especially, of the distance) was not anti-monotonic which prevented the computation of spatio-temporal patterns during the mining step. Moreover, the spatio-temporal patterns obtained were quite small in practice which led to a low recall when using them for object tracking. In this article, we present an extended version of the plane graph mining algorithm called DYPLAGRAM_ST which can benefit from the spatio-temporal constraints to directly and thus more efficiently mine the spatio-temporal patterns. Besides, we propose a method based on a global occurrences graph to combine these patterns in order to build spatio-temporal paths that can be used to follow some objects in the videos. By allowing a pattern to change along a path, it is possible to take into account instability in the video or change of view point which improves the recall of the patterns.

The tracking methods presented at the beginning of this introduction typically do not consider moving objects in changing environments. When it is the case as in [4], multiple cameras are used to tackle object occlusions or features instability using stereo vision. The setting taken in [8] is close to the one we are interested in since they consider cameras embedded in surveillance cars but they rely on strong background information (here GPS position) to perform an effective tracking. Our method is also similar to [18] but they do not use the topological information provided by the subgraph patterns and they use a spatio-temporal Markov Chain Monte Carlo algorithm to sample the possible paths represented in our occurrences graph.

The outline of this paper is the following. In Section 2, we recall some important definitions and explain the proposed extensions to the DyPlagram algorithm proposed in [14]. In Section 3 we show how to compute the spatio-temporal paths used for object tracking. Section 4 shows a large set of experiments on a synthetic and on a real video to assess both the efficiency of our new algorithm DyPlagram_st but also the usefulness of the spatio-temporal paths to tackle the problem of object tracking in videos. We conclude in Section 5.

2 Spatio-temporal Patterns Mining

2.1 Dynamic Plane Graphs

The definitions in this section are similar to those of [14]. As in [14], our algorithm mines 2-connected plane graphs that satisfy various spatio-temporal constraints. The restriction to 2-connected plane graphs was motivated by the use of plane graphs in our video data and because it allows to test subgraph isomorphism in polynomial time. Moreover, this restriction also dramatically decreases the branching factor of the search space and improves the efficiency (as already shown in [14]).

Definition 1 (Plane graph). *A plane graph is $G = (V, E, F, f_e, L)$ where V is a set of nodes, E is a set of edges, F is a set of faces and L is a labeling function on $V \cup E$. Exactly one of the faces $f_e \in F$ is called the external face, the other faces are the internal faces. The graph is 2-connected if each face is a simple cycle (the face does not use a node or an edge more than once).*

Our aim is to find 2-connected plane subgraphs which satisfy some constraints in a database of graphs.

Definition 2 (Plane subgraph isomorphism, occurrence). *Given two plane graphs $G = (V, E, F, f_e, L)$ and $G' = (V', E', F', f'_e, L')$, G' is a* plane subgraph *of G if there is an injective function f from V to V' which preserves the edges, the internal faces of G' and the labels. The function f is called an* occurrence *of G' in G.*

The frames in a video are ordered, and this order is taken into account when computing spatio-temporal patterns. We thus define a dynamic graph as an ordered set of graphs.

Definition 3 (Dynamic plane graph). *A dynamic plane graph $\mathcal{D}$ is an ordered set of plane graphs $\{G_1, G_2, .., G_n\}$. Each node of these graphs is associated to spatial coordinates (x, y).*

Example 1. In our video application, each plane graph G_i represents a video frame. Each node in a graph represents a segmented frame region, and is associated to the coordinates (x, y) of the barycenter of this region. The labels on nodes are built either by a discretization of the size of the regions or of the color.

We define an occurrence of a plane graph in a dynamic plane graph and its frequency.

Definition 4 (Occurrences of a plane graph in a dynamic graph). *Given a plane graph P and a dynamic graph $\mathcal{D} = \{G_1, ..., G_n\}$, the set of occurrences of P in $\mathcal{D}$ is defined as $Occ(P) = \{(i, f) \mid f$ is an occurrence of P in $G_i\}$.*

Definition 5 (Frequency of a plane graph in a dynamic graph). *The frequency $freq(P)$ of a plane graph P in a dynamic graph $\mathcal{D}$ is the number of graphs $G_i \in \mathcal{D}$ in which there is an occurrence of P, i.e., $|\{i \mid \exists f,\ (i, f) \in Occ(P)\}|$.*

2.2 Occurrences Graph and Spatio-temporal Patterns

In typical subgraph mining problems, where the input collection of graphs does not represent a dynamic graph, the frequency $\text{freq}(P)$ of a pattern graph P is computed regardless of the fact that its occurrences may be far apart w.r.t. time and/or space. To define a frequency that takes into account spatio-temporal distance between the occurrences, we define in this section the notion of an occurrences graph in which occurrences of the same pattern that are close to one another are linked. Then, we define spatio-temporal patterns in this occurrences graph and the associated frequency (called freq_{st}).

The definitions in this section, although similar to the one of [14], have been changed to integrate the spatio-temporal patterns computation during the mining step instead of during a post-processing step. This offers more pruning opportunities.

Definition 6 (Distance between occurrences). *The distance between two occurrences $o = (i, f)$ and $o' = (i', f')$ of a plane graph $P = (V, E, F, f_e, L)$ in a dynamic graph $\mathcal{D}$ is defined as: $dist(o, o') = \max_{s \in V} d(f(s), f'(s))$, where d denote the Euclidean distance between the nodes.*

This distance has an anti-monotonic property:

Proposition 1. *For any patterns $P = (V, E, F, f_e, L)$ and $P' = (V', E', F', f'_e, L')$ such that P is a plane subgraph of P' and two occurrences $o_1 = (f_1, i)$, $o_2 = (f_2, i)$ of P and two occurrences $o'_1 = (f'_1, i)$, $o'_2 = (f'_2, i)$ of P' such that f_1 is a restriction of f'_1 (i.e., $f_1 = f'_1$ on V) and f'_2 is a restriction of f_2, then we have $dist(o_1, o_2) \leq dist(o'_1, o'_2)$.*

proof (sketch): the set from which the maximum is taken for $\text{dist}(o_1, o_2)$ is included in the set for which the maximum is taken for $\text{dist}(o'_1, o'_2)$.

The depth first traversal of the search space by our mining algorithm define a parent relationship on patterns:

Definition 7 (Parent of a pattern and of an occurrence). *Given a pattern P with $n \geq 2$ internal faces, the pattern $p(P)$ with $n-1$ faces from which P was built is called the parent of P. And given an occurrence $o = (f, i)$ of P, we call the* parent of o *the occurrence $p(o) = (f', i)$ such that f' is the restriction of f to the nodes of $p(P)$.*

The definition of the parent of an occurrence is then used to define the occurrences graph. The nodes of the occurrences graph are the occurrences of a pattern and the edges connect "close" occurrences. This graph is constructed for each pattern in the mining algorithm.

Definition 8 (Occurrences graph and Spatio-temporal pattern). *Given a spatial threshold ϵ, a temporal threshold τ, a plane graph $P = (V, E, F, f_e, L)$ and a dynamic graph $\mathcal{D}$, we define the* occurrences graph of P *as an oriented graph whose set of nodes is $Occ(P)$.*

- *If P has only one face, then there is an edge from (f, i) to (g, j) if $0 < j - i \leq \tau$ and $dist(f, g) \leq \epsilon.(j - i)$ and there is no (h, k) with $i < k < j$ and $dist(f, h) \leq \epsilon.(k - i)$.*
- *If P has more than one face, then there is an edge from $o = (f, i)$ to $o' = (g, j)$ if there is an edge $(p(o), p(o'))$ in the occurrences graph of $p(P)$ and $dist(f, g) \leq \epsilon.(j - i)$.*

A spatio-temporal pattern S based on P *is a connected component of the occurrences graph of P.*

This definition is such that the occurrences graph of a pattern P is always a subgraph of the occurrences graph of its parent pattern $p(P)$ (if we identify the node o of the occurrences graph of P with the node $p(o)$ of the occurrences graph of $p(P)$). This ensures that the spatio-temporal patterns based on P get "smaller" as the pattern P grows, and this ensures that the frequency of a spatio-temporal pattern defined below has the anti-monotonicity property.

Definition 9 (Frequency of a spatio-temporal pattern). *The frequency of a spatio-temporal pattern S based on a graph pattern P in a dynamic graph $\mathcal{D}$ is $freq_{st}(S) = |\{i \mid \exists f,\ (i, f) \in S\}|$.*

Example 2. Fig. 1 shows 11 occurrences of a pattern P in a video with five frames. $\text{freq}(P) = 5$. Since occurrences 1 and 4 are close to each other, i.e., their spatial distance is lower than 2ϵ and their temporal distance is $2 \leq \tau$, there

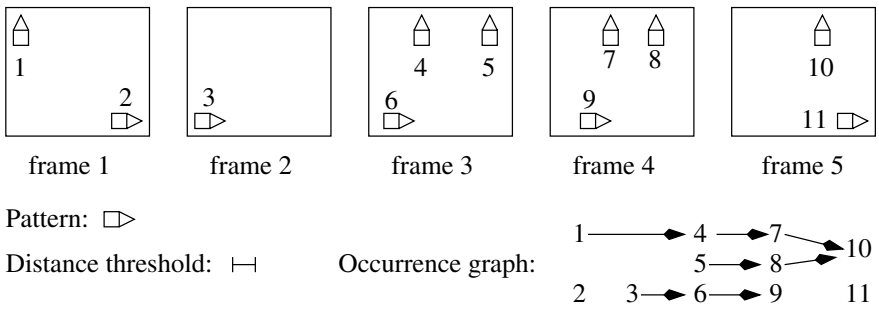


Fig. 1. Occurrences of a pattern and occurrences graph of this pattern (temporal threshold $\tau = 2$ and distance threshold ϵ)

is an edge $(1, 4)$ in the occurrences graph of P. Conversely, the edges $(3, 5)$ or $(2, 11)$ do not exist in the occurrences graph, as the spatial distance between 3 and 5 or the temporal distance between 2 and 11 are too large. There are 4 spatio-temporal patterns $S_1 = \{1, 4, 5, 7, 8, 10\}$, $S_2 = \{3, 6, 9\}$, $S_3 = \{2\}$ and $S_4 = \{11\}$. The frequencies of these patterns are: $\text{freq}_{st}(S_1) = 4$, $\text{freq}_{st}(S_2) = 3$, and $\text{freq}_{st}(S_3) = \text{freq}_{st}(S_4) = 1$.

Proposition 2. *Given a pattern P with more than one face, and given a spatio-temporal pattern S based on P then there is a spatio-temporal pattern S' based on the parent $p(P)$ of P with a larger freq_{st}, i.e., $\text{freq}_{st}(S) \leq \text{freq}_{st}(S')$.*

This proposition shows that, given a minimum threshold minfreq_{st} on freq_{st}, if a pattern does not have a frequent spatio-temporal pattern then any super-pattern does not either. This allows to prune the search space.

2.3 DyPlagram_st Algorithm

Given a frequency threshold minfreq (also called minimum support), a minimum threshold minfreq_{st} for freq_{st} a spatial threshold ϵ and a temporal threshold τ, the proposed algorithm DyPlagram_st computes all spatio-temporal patterns with $\text{freq}_{st} \geq \text{minfreq}_{st}$ based on patterns with $\text{freq} \geq \text{minfreq}$ (the thresholds ϵ and τ are used in the construction of the occurrences graph, see Def. 8).

The proposed algorithm DyPlagram_st is based on DyPlagram [14] which itself is based on gSpan. Its main characteristics are :

- a recursive depth first exploration of the search space;
- the use of canonical codes to avoid considering the same graph several times;
- at each level, patterns are extended by adding a whole face to the current pattern.

The new definition of the freq_{st} is now anti-monotonic, and we can use it in the DyPlagram_st algorithm. However, this frequency is not defined on patterns but on spatio-temporal patterns. We must therefore also build the occurrences graph and the spatio-temporal patterns in the algorithm.

Given an occurrence $o = (f, i)$ of a pattern P, an extension E of P is a set of edges such that $P \cup E$ has exactly one more face than P and there is an occurrence $o' = (f', i)$ of $P \cup E$ that extends o, i.e., such that f is the restriction of f' to P.

As its predecessors, DyPlagram_st uses canonical codes to represents patterns and extensions. This allows to efficiently enumerate only the so called valid extensions of a pattern. Informally, a valid extension of a pattern is an extension that lead to a pattern not already considered by the algorithm. This is a very efficient way to avoid considering several times the same pattern. We do not detail here how these codes are built, the interested reader can refer to [14].

The DyPlagram_st algorithm first builds all frequent one face patterns and then calls the following recursive function `mine` for all of them.

mine$(P, \text{minfreq}, \text{minfreq}_{st}, \tau, \epsilon, \mathcal{D})$

```
1   occurrences_graph(P) = empty_graph
2   for each occurrence of P in D do
3       Add this occurrence to occurrences_graph(P)
4       Computes all valid extensions of this occurrence
5   Computes the edges of occurrences_graph(P) (using ε and τ)
6   Computes all spatio-temporal patterns based on P
7   for each spatio-temporal pattern S based on P do
8       if freq_st(S) ≥ minfreq_st then output(S)
9   if there is no frequent spatio-temporal pattern then return
10  else
11      for each extension E of P do
12          if the code of E ∪ P is canonical and freq(E ∪ P) ≥ minfreq then
13              mine(P ∪ E, minfreq, minfreq_st, τ, ε, D)
14      return
```

In this algorithm, lines 1, 3, 5, 6, 7, 8, and 9 were not in DyPlagram [14].

Thanks to Prop. 2, this algorithm is correct and output exactly the spatio-temporal patterns whose freq_{st} is above the user defined threshold σ.

3 Spatio Temporal Path

When tracking an object in a real video, we cannot expect that the object is represented by the same graph pattern during the whole video (e.g., due to changes in view point or instability of the segmentation). Thus, if we want to track it using spatio-temporal patterns, we propose to build a path in the union of all occurrences graphs. To allow this path to "jump" from a spatio-temporal pattern to another, similarity edges are added between overlapping occurrences of different patterns. Weights are also added on the edges so that minimum weight paths can then be computed in this global occurrences graph.

Definition 10 (Similarity of two occurrences). *Let $o = (i, f)$ and $o' = (i, f')$ be two occurrences of two different patterns $P = (V, E, F, f_e, L)$ and $P' = (V', E', F', f'_e, L')$. The similarity between these occurrences is defined as $\sigma(o, o') = \frac{|f(V) \cap f'(V')|}{|f(V)|}$.*

This similarity is not symmetric and it is used to weight the edges in the global occurrences graph.

Definition 11 (Global occurrences graph). *Given a set of patterns $\mathcal{P}$, temporal and spatial thresholds τ and ε, a similarity threshold σ, the global occurrences graph is a weighted oriented graph: its node set is $V = \cup_{P \in \mathcal{P}} Occ(P)$ and its edge set is $E = E_{\mathcal{P}} \cup E_{sim}$ where :*

- *$E_{\mathcal{P}}$ is the union of the edge sets of all patterns occurrences graphs. The weight of an edge $((i, f), (i', f'))$ is $w = \frac{(i' - i - 1)}{\tau}$.*

– $E_{sim} = \{(o, o', w) |\ o = (i, f), o' = (i, f'), \sigma(o, o') < \sigma\}$ *is the set of similarity edges with*

$$w = \begin{cases} 0 & \text{if } |V| < |V'| \\ \frac{1}{2}(\frac{1-\sigma(o,o')}{1-\sigma} + \frac{d}{\epsilon}) & \text{otherwise.} \end{cases}$$

where V and V' are the node sets of the patterns corresponding resp. to occurrences o and o', and d is the distance between the barycenters of o and o'.

A spatio-temporal path *is a path in the global occurrences graph.*

The edges in $E_{\mathcal{P}}$ are edges between 2 occurrences of the same pattern that are not in the same frame. If these two occurrences are in consecutive frame, the weight is 0 (when $i' = i + 1$) otherwise the weight increases with the number of frames between them (normalized by the temporal threshold τ).

The edges in E_{sim} are *similarity edges* between 2 occurrences of different patterns that are in the same frame and whose similarity is below σ. We want to favor paths that use large patterns, thus the weight of an edge from an occurrence of a small pattern to a larger one is 0. The weight of an edge from an occurrence of a large pattern to a smaller one increases as the similarity decreases and the spatial distance increases.

4 Experiments

Some experiments in [14] have already assessed the efficiency of the plane graph mining algorithm called DyPlagram compared to a generic graph mining algorithm such as gSpan [16]. The introduction of this plane graph mining algorithm was necessary to effectively mine the graphs extracted from videos. Experiments on a very simple video (one object, moving background, no occlusion, no disappearance) showed promising results for object tracking but the interest of the spatio-temporal patterns for more complex videos was not thoroughly evaluated. Besides, in [14], we did not present an effective way to use spatio-temporal constraints in DyPlagram nor a systematic method to combine the spatio-temporal patterns into spatio-temporal paths to track object in videos. Our proposed experiments aim to answer three main questions:

1. Are the spatio-temporal constraints well exploited by the new DyPlagram_st algorithm compared to the process presented in [14]?
2. How are the results of the DyPlagram_st algorithm where the spatial and the temporal constraints are pushed directly into the mining process compared to the post-processing experiments described in [14]?
3. How meaningful (in terms of precision and recall) are the spatio-temporal paths to track objects in a synthetic and in a real video?

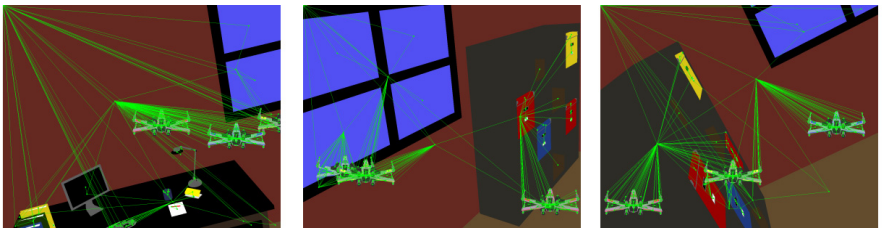


Fig. 2. Example of RAGs obtained from the synthetic video

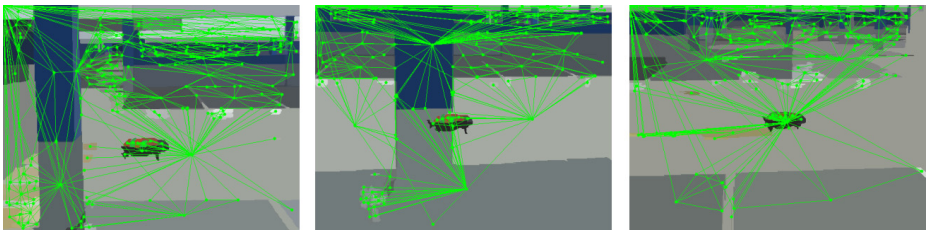


Fig. 3. Example of RAGs obtained from the real video

4.1 Video Datasets

We used 2 datasets for these experiments. One was created from a synthetic video which allows us to avoid the possible segmentation problems. The second comes from a real (but simple) video with its possible segmentation issues.

For both videos, we used two possible labels on the nodes of the RAGs. The first possible one comes from a *discretization of the size* of the segmented regions (in pixels). The discretization uses 10 bins of equal size that were computed using all the possible region sizes (sorted for the discretization) for a given video. The second is a *color discretization* of the mean color of the segmented regions. We divided each of the 3 RGB channels in 3 parts, resulting in 27 bins of equal range.

The synthetic video has 721 frames in total. In average the RAGs are composed of 240.7 nodes with an average degree of 3.9. Three identical objects (X-wings) are moving in the video such that they may overlap or even get (partially) out of the field of view (this helped us to evaluate how well spatio-temporal patterns can be used to represent the trajectory of the X-wings individually). The 3 X-wings have different colors but this feature is not always used in the experiments. Fig. 2 show three examples of RAGs we obtained for this dataset.

The real video is composed of 950 frames (25 frames per second), each RAG has on average 194.5 nodes with an average degree of 5.35. This video shows a drone flying across a covered parking lot. Before building the RAGs, we segmented each frame of the video independently using the algorithm presented in [10] and available on the web[1]. This algorithm has 3 parameters for which we

[1] http://www.cs.brown.edu/~pff/segment/

used standard values. This algorithm helps the merging of small regions which may result in an unstable segmentation when objects are getting close to or moving away from the camera. In order to prevent this behavior, we modified the code of this algorithm to make its second parameter independent from the size of the regions. Fig. 3 show three examples of RAGs we obtained for this video.

4.2 Evaluation of the Patterns

To evaluate our spatio-temporal patterns, we use some ground truth. For both the real and the synthetic videos, we have tagged the positions of the plane(s) (objects o) in each frame of the video.

We introduce two measures which assess how precisely a spatio-temporal pattern p corresponds to a given target object o in the video frames. These measures are adaptations of the popular *precision* and *recall* measures as described below:

- **precision**: fraction of the occurrences of p (in the target graphs) of which every node maps to o in the corresponding video frames. The intuition behind this measure is to evaluate the *purity* of p, that is, p has the maximum precision if it maps only to o and nothing else.
- **recall**: Let n be the number of frames in which o is present. The recall is defined as the fraction of n in which there exists at least one occurrence of p where every node maps to o. Here, the intuition is to evaluate the *completeness* of p. More precisely, the idea is to check whether the occurrences of p map to all occurrences of o in the set of video frames.

Since our algorithm is exhaustive, that is, it mines for all frequent spatio-temporal patterns in the graph database without supervision, the mining result may consist of different spatio-temporal patterns corresponding to different objects, or even to no specific one (w.r.t. the proposed measures). To be able to evaluate the precision and recall of our spatio-temporal patterns for all 3 different planes (in the synthetic video) and for the drone (in the real video), we have considered that the spatio-temporal patterns starting in every frames of the video have been tagged according to the object it belongs to. In other words, we evaluate the precision and recall of each spatio-temporal patterns knowing in advance what the first occurrence of the pattern in each occurrences graph maps to.

4.3 Spatio-temporal Paths for Object Tracking

To assess the effectiveness of the spatio-temporal paths for object tracking, we apply the following strategy. We first build the occurrences graph and then, for each target object, we select the occurrences matching them in the first frame and compute the path of lowest cost starting from those occurrences and reaching the last frame using Dijkstra's shortest path algorithm. In all experiments reported here we use a similarity of $2/3$ ($\sigma = 0.65$).

Table 1. Evaluation of the connected components (CC) issued from all patterns with minfreq = 721 and $\tau = 1$ for DyPlagram and for DyPlagram_st. The labels are created from the size of the region.

	DyPlagram with post-processing			DyPlagram_st		
	$\epsilon = 10$, $\text{minfreq}_{st} = 10$					
	Precision(%)	Recall(%)	Number of CCs	Precision(%)	Recall(%)	Number of CCs
plane 1	78	7	151	78	7	**114**
plane 2	72	3	129	**95**	**3**	71
plane 3	87	2	**131**	88	2	**84**
	$\epsilon = 20$, $\text{minfreq}_{st} = 50$					
	Precision(%)	Recall(%)	Number of CCs	Precision(%)	Recall(%)	Number of CCs
plane 1	77	15	73	82	17	65
plane 2	93	26	43	**100**	29	**39**
plane 3	100	10	60	100	10	60
	$\epsilon = 170$, $\text{minfreq}_{st} = 50$					
	Precision(%)	Recall(%)	Number of CCs	Precision(%)	Recall(%)	Number of CCs
plane 1	45	38	27	**51**	42	24
plane 2	51	10	15	49	8	17
plane 3	60	12	21	**69**	13	19

In practice the minimum support threshold minfreq can be set, for example, to 1/5 of the total number of frames (to make sure that the patterns occur enough and help the mining process). By default, it will be equal to the minfreq_{st} threshold. minfreq_{st} should be set as low as possible (depending on available memory). The τ should, in general, be set as high as possible (as will be shown in the experiments). The ϵ constraint depends on the motion speed of the target object and on the resolution of the video. We most of the time use 20 pixels.

4.4 DyPlagram_st vs DyPlagram

The experiments showed in Table 1 allow us to compare the DyPlagram algorithm presented in [14] with our new upgraded algorithm, DyPlagram_st, which uses the spatial and the temporal constraints directly in the mining process. These experiments are made with the same synthetic video as in [14] and the same discretization procedure which uses only the size of the region. The same minimum support (minfreq = 721) has been used as well as the same *gap* constraint $\tau = 1$ as in [14]. The minfreq_{seq} threshold used in DyPlagram to prune part of the search space is not used by DyPlagram_st which uses a different minfreq_{st} threshold (explained in Sec. 2.2). However, in these experiments, we set the same threshold for freq_{seq} and freq_{st}. Note that the results for DyPlagram are not exactly the ones reported in [14] because we found that the strategy proposed in [14] was overly optimistic as far as precision was concerned. Indeed, the chosen spatio-temporal patterns (i.e., the connected components (CC)) were the ones for which the first occurrence matched a pattern that was selected in the first frame of the video. This means that a spatio-temporal pattern that also matched a chosen object but for which the first occurrence belongs to a pattern that was not selected in the first frame would not be taken into account to compute the precision of this object. Here we compute the precision and recall for all the CC whose first occurrence matches an object of interest. We expect the precision/recall results to be comparable for both algorithms, although the CC

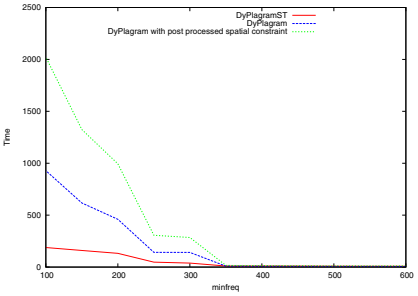


Fig. 4. Time(s) taken by both versions of the DyPlagram algorithm, with $\tau = 1$, $\text{minfreq}_{st} = 50$ and $\epsilon = 20$ to generate all the occurrences (red vs blue line) and to generate the occurrences graph ((red vs green line)

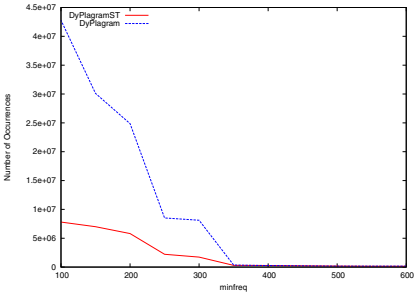


Fig. 5. Number of occurrences generated by DyPlagram while pushing the spatial constraint (red plain line) or not (blue dashed line)

computed by DyPlagram are expected to be more numerous than the ones computed by DyPlagram_st.

As can be seen in Table 1, the connected component obtained with DyPlagram_st are in general less numerous, more precise and have a better recall than the ones obtained with DyPlagram. As already discussed in [14], the distance threshold ϵ has an important impact on the obtained results. Indeed, if it is set too low (to 10 pixels, in our example), we obtain spatio-temporal patterns with high average precision for each X-wing as different occurrences of patterns which map to different X-wing are very well distinguished. However, this leads to a low average recall: since only very close occurrences of the same pattern are linked, the spatio-temporal patterns tend to be short (i.e., have low freq_{st}). When using a distance threshold $\epsilon = 10$, no spatio-temporal patterns with $\text{freq}_{st} \geq 50$ were found for X-wing2 for DyPlagram ([14]), which explains why we used a minfreq_{st} of 10 in this case. Conversely, for a higher ϵ of 170 pixels, the average precision drops as the different X-wings are not well distinguished anymore. For example, it was possible to obtain spatio-temporal patterns with higher recall for the plane 1 (when comparing to the other experiments), but, they had low average precision. Since the plane 1 gets partially out of the video frames around 6 times, a higher number of spatio-temporal patterns were derived for this X-wing for $\text{minfreq}_{st} = 50$ and ϵ of at least 20, which represent the different time intervals where this X-wing is visible through the video. As another example, the plane 2 is hidden only twice by the plane 3 (during around 15 frames) and never goes out of the video frames. This explains the lower number of patterns found for this object, also for $\text{minfreq}_{st} = 50$ and $\epsilon \geq 20$.

Fig. 4 and 5 show efficiency results comparing DyPlagram [14] and DyPlagram_st. As expected, pushing the spatial constraints during the mining step allow us to generate less occurrences (especially for support < 350) in a lower time.

Table 2. Evaluation of the spatio-temporal path with minfreq = 250, $\text{minfreq}_{st} = 150$, $\sigma = 0.65$, $\epsilon = 20$. The numbers between parenthesis correspond to the best precision and recall of the best path in term of recall, and the emphasized results are the best results for each plane

		Size Discretization			Color Discretization		
	τ	Precision(%)	Recall(%)	Paths	Precision(%)	Recall(%)	Paths
plane 1	10	**98.32** (99.72)	**97.50** (99.30)	34	93.92 (99.74)	93.60 (99.86)	21
plane 2		**99.63** (99.73)	**97.26** (98.19)	24	98.65 (100)	**96.82** (99.02)	17
plane 3		**9.49** (16.64)	**8.70** (15.39)	4	- (-)	- (-)	0
plane 1	25	95.79 (100)	94.59 (99.02)	38	**99.17** (99.73)	**98.40** (100)	21
plane 2		65.66 (99.61)	64.61 (98.05)	32	98.54 (100)	96.34 (99.02)	20
plane 3		2.93 (9.09)	2.50 (8.59)	29	31.95 (31.95)	29.54 (29.54)	2
plane 1	100	79.05 (100)	74.37 (94.31)	42	97.76 (100)	95.36 (99.30)	29
plane 2		72.57 (97.53)	67.05 (93.62)	35	**98.87** (100)	96.30 (99.02)	39
plane 3		5.42 (18.46)	4.82 (16.36)	31	**86.27** (90.52)	**75.92** (82.80)	23

4.5 Evaluation of the Spatio-temporal Path for Object Tracking

For both datasets, we report the precision and recall results for the spatio-temporal patterns (which have a first occurrence on the object of interest anywhere in the video) and for the spatio-temporal paths (which have a first occurrence on the object of interest in the first frame of the video). The spatio-temporal patterns or connected components (CC) correspond to the global occurrences graph without the similarity edges.

Synthetic Video. The experiments reported in Table 2 show the precision and recall results for the paths obtained on the synthetic video when varying the gap between 10 and 100. Results for the CC are similar to the ones reported in Table 1.

Because of the nature of the video, we use a global minimum support minfreq of 250 in order to prune the number of frequent patterns. Indeed, since the synthetic video has been especially made to produce stable graphs, DyPlagram_st returns a lot of frequent patterns on this dataset which leads to a huge global occurrences graph that possibly does not fit into memory for processing. To be able to perform various experiments, especially with the size discretization which does not permit to distinguish the three planes at the mining step, we set the minfreq_{st} to 150 (although as already discussed, it is better to set it as low as possible).

Overall, we obtain very good results for the first two planes (precision and recall close to 100%). We can clearly see the lack of discriminative power of the size discretization when the gap increases. Indeed the paths start to follow different planes, reducing their precision and their recall. For those two planes the color discretization always shows good results, with average precisions and recalls close to the ones of the best paths (values in brackets). Since the 3rd plane moves back and forth horizontally across the field of view (getting almost completely out every 120 frames), only few paths starting on the plane manage to reach the end of the video when we use a low gap. The paths which uniquely follow this plane are thus more expensive than other paths on which the algorithm can "jump" using the similarity edges decreasing the precision and recall. As

Table 3. Precision, recall and coverage recall computed for the connected components computed and for the real video with minfreq = minfreq_{st}, and $\sigma = 0.65$

		$\epsilon = 10$			$\epsilon = 20$		
τ	minfreq_{st}	Precision(%)	Recall(%)	CC	Precision(%)	Recall(%)	CC
10	100	**100**	26.18	10	**92.48**	22.97	13
	50	93.55	17.40	20	91.35	15.44	25
	10	89.78	2.87	294	89.70	2.72	334
25	100	91.28	35.34	11	89.02	30.03	14
	50	90.28	25.12	18	83.79	20.14	24
	10	88.90	3.18	307	89.47	2.94	358
100	100	89.52	**38.21**	14	89.02	**31.03**	19
	50	92.27	24.38	27	90.30	22.45	30
	10	89.01	4.03	258	89.88	3.63	302

we can see, increasing the gap allows to overcome this problem with the color discretization while keeping good results for the other two planes.

Real Video. The experiments reported in Table 3 and 4 were made without using a global minimum support threshold (which is equivalent to set minfreq = minfreq_{st}). Because of the segmentation, this dataset is a lot less stable than the synthetic one resulting in less frequent patterns. For this one, so far, only the color discretization gave good precision/recall results (we also tried the size and some other color discretization).

Connected Component (CC). The results for the connected components are presented in Table 3. Those experiments have been obtained for $\epsilon = 10$ and $\epsilon = 20$, above that the precision started to drop significantly (which is expected for large ϵ values if other distracting objects are frequent).

As expected, the precision is a little higher with $\epsilon = 10$ (100% for $\epsilon = 10$ when $\tau = 10$ and $\text{minfreq}_{st} = 100$ against 92.48% for $\epsilon = 20$). The fact that the average recall also decreases with a higher distance is more surprising at first glance. This is explained by the fact that most of the time, $\epsilon = 10$ is enough to follow the drone, but sometimes the drone or the camera movement accelerates. In those cases a higher distance might give longer and better CC but also might introduce some noisy ones which would decrease the average recall and precision.

The average recall also lowers when we lower minfreq_{st}. This is due to the fact that when using a low minfreq_{st} DyPlagram_st outputs short spatio-temporal patterns that necessarily have a low recall. Lowering minfreq_{st} slightly reduces the precision of the connected components but increases their number.

As also expected, higher gaps lead to better recall (38.21% for $\tau = 100$ when $\epsilon = 10$ and $\text{minfreq}_{st} = 100$ against 26.18% for $\tau = 10$) as well as improve the coverage of the spatio-temporal patterns in the whole video. The precision doesn't seem to be influenced by τ when we allow small spatio-temporal patterns (i.e., a low minfreq_{st}).

Spatio-temporal Paths. Table 4 shows the results for the CC on the real dataset for the color discretization.

A distance ϵ equal to 20 gives the best results in most cases with high precision and good recall (99.03 for precision and 80.63 for the recall for $\epsilon = 20$, $\tau = 25$

Table 4. Precision and recall computed for the spatio-temporal paths for the real video with minfreq = minfreq_{st} and $\sigma = 0.65$

		$\epsilon = 10$			$\epsilon = 20$		
τ	minfreq_{st}	Precision(%)	Recall(%)	Paths	Precision(%)	Recall(%)	Paths
10	100	96.30 (96.30)	67.89 (67.89)	1	98.23 (100)	80.94 (82)	2
	50	98.25 (100)	70.00 (71.26)	2	26.16 (38.96)	24.03 (36.21)	3
	10	91.93 (93.34)	69.60 (70.63)	8	18.75 (36.09)	17.88 (34.73)	8
25	100	98.43 (100)	68.89 (70)	6	98.51 (100)	78.68 (79.68)	6
	50	98.66 (100)	69.05 (70)	7	98.72 (100)	78.82 (79.68)	7
	10	99.06 (100)	69.36 (70.21)	10	**99.03 (100)**	**80.63 (81.36)**	10
100	100	100 (100)	67.42 (67.78)	8	100 (100)	77.52 (79.68)	9
	50	100 (100)	67.36 (67.68)	9	100 (100)	77.54 (79.68)	9
	10	100 (100)	67.21 (67.78)	10	99.26 (100)	79.17 (79.78)	10

and $\text{minfreq}_{st} = 10$ for example). However, the values for $\tau = 10$ show the limits of the use of the shortest path algorithm to tackle our problem. Similarly to what was happening with the third plane in the synthetic video, the shortest path might not always be following the object we want to track if elements in the background or other objects offer better stability than the object we want to track and are close enough to "jump" on them.

The results with our preferred setting (low $\text{minfreq}_{st} = 10$, high $\tau = 100$ and a distance $\epsilon = 20$) show that the spatio-temporal paths can indeed be used to follow an object in the video. The similarity edges introduced are very useful to increase the recall of the patterns and experiments with a higher similarity constraint (for example with $\sigma = 0.8$) show worst results. This shows the importance of this "inexact" matching phase in the process. On the downside, the choice of the labels on the node (here it is a color information) seems to play a very important role to get interesting spatio-temporal patterns although it is difficult to evaluate in an unsupervised setting what could be the best ones. One solution could be to attach more diverse information on the labels of the nodes to overcome this problem.

5 Conclusion

We have presented an unsupervised method based on graph mining to track objects in videos. More precisely, we have used paths computed in an occurrences graph of these frequent graph patterns. The graph is created by linking through spatial, temporal and similarity constraints the frequent patterns to follow one or multiple objects simultaneously in a video. The results on a synthetic and on a real video show that this method is effective to tackle our tracking problem. However, it strongly relies on the labels of the nodes (discretization and chosen features). This problem could be tackled by taking into account multiple and diverse ordered information on the nodes to automatically select the best features depending on the video. Some future work could also be done on the computation of the best paths in the video as the current shortest path algorithm assumes that our objects of interest are followable from the first to the last frame of the video. Although still naive, we believe that our method could be useful to tackle the difficult problem of tracking multiple objects in the specific case in which both

the objects and the background are moving and when no supervised information about the objects to track is known in advance. The proposed method could also benefit from the very recent work which uses supporters (points or objects moving in a correlated way with the tracked objects) or distracters (objects which should not be confused with the objects to track) for example presented in [9] as these would typically represent correlated frequent subgraphs.

References

1. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys 38(4), 13+ (2006)
2. Berlingerio, M., Bonchi, F., Bringmann, B., Gionis, A.: Mining Graph Evolution Rules. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part I. LNCS, vol. 5781, pp. 115–130. Springer, Heidelberg (2009)
3. Borgwardt, K.M., Kriegel, H.P., Wackersreuther, P.: Pattern mining in frequent dynamic subgraphs. In: Proceedings ICDM. pp. 818–822 (2006)
4. Cai, L., He, L., Xu, Y., Zhao, Y., Yang, X.: Multi-object detection and tracking by stereo vision. Pattern Recogn. 43(12), 4028–4041 (2010)
5. Celik, M., Shekhar, S., Rogers, J.P., Shine, J.A.: Mixed-drove spatiotemporal co-occurrence pattern mining. IEEE TKDE 20(10), 1322–1335 (2008)
6. Chang, R.F., Chen, C.J., Liao, C.H.: Region-based image retrieval using edgeflow segmentation and region adjacency graph. In: IEEE ICME, pp. 1883–1886 (2004)
7. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. 24(5), 603–619 (2002)
8. Diego, F., Evangelidis, G., Serrat, J.: Night-time outdoor surveillance by mobile cameras. In: ICPRAM (2012)
9. Dinh, T.B., Vo, N., Medioni, G.G.: Context tracker: Exploring supporters and distracters in unconstrained environments. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1177–1184 (2011)
10. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. Int. J. Comput. Vision 59(2), 167–181 (2004)
11. Goszczynska, H.: Object Tracking. InTech (2011)
12. Kim, Z.: Real time object tracking based on dynamic feature grouping with background subtraction. In: IEEE CVPR (2008)
13. Kuo, C.H., Huang, C., Nevatia, R.: Multi-target tracking by on-line learned discriminative appearance models. In: IEEE CVPR, pp. 685–692 (2010)
14. Prado, A., Jeudy, B., Fromont, E., Diot, F.: Mining spatiotemporal patterns in dynamic plane graphs. IDA Journal 17(1) (to appear, 2013)
15. Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: IEEE CVPR, pp. 2432–2439 (2010)
16. Yan, X., Han, J.: Gspan: Graph-based substructure pattern mining. In: IEEE ICDM, pp. 721–724 (2002)
17. Yang, H., Parthasarathy, S., Mehta, S.: A generalized framework for mining spatiotemporal patterns in scientific data. In: ACM SIGKDD, pp. 716–721 (2005)
18. Yu, Q., Medioni, G.: Multiple-target tracking by spatiotemporal monte carlo markov chain data association. IEEE Trans. Pattern Anal. Mach. Intell. 31(12), 2196–2210 (2009)

Hypergraph Learning with Hyperedge Expansion

Li Pu and Boi Faltings

Artificial Intelligence Laboratory
École Polytechnique Fédérale de Lausanne
CH-1015, Lausanne, Switzerland
{li.pu,boi.faltings}@epfl.ch

Abstract. We propose a new formulation called *hyperedge expansion* (HE) for hypergraph learning. The HE expansion transforms the hypergraph into a directed graph on the hyperedge level. Compared to the existing works (e.g. star expansion or normalized hypergraph cut), the learning results with HE expansion would be less sensitive to the vertex distribution among clusters, especially in the case that cluster sizes are unbalanced. Because of the special structure of the auxiliary directed graph, the linear eigenvalue problem of the Laplacian can be transformed into a quadratic eigenvalue problem, which has some special properties suitable for semi-supervised learning and clustering problems. We show in the experiments that the new algorithms based on the HE expansion achieves statistically significant gains in classification performance and good scalability for the co-occurrence data.

1 Introduction

Many tasks require clustering in a graph where each edge represents a similarity relation. Often, it is a co-occurrence relation that involves more than two items, such as the co-citation and co-purchase relations. The co-occurrence relation can be represented by a hyperedge that connects two or more vertices in a hypergraph. But most clustering algorithms, such as k-means, or spectral clustering, are defined for graphs but not hypergraphs. Therefore, hyperedge relations are often transformed into another graph that is easier to handle [1,2,3].

For classification and clustering tasks, the hyperedges are usually transformed into cliques of edges. This category of techniques includes *clique expansion*, *star expansion* [4], and *normalized hypergraph cut* (NHC) [5]. In Figure 1 we shown a simple example of such transformation from a hypergraph to a graph (the *induced graph*). Since the transformations are carried out on the vertex level, we call them *vertex expansions*.

With a vertex expansion, evaluating the goodness of clustering is done on the induced graph. For example, in a hyperedge of k vertices, a cut that separates the hyperedge into 1 and $k-1$ vertices would cut $k-1$ pairwise edges, while a cut that splits the vertices in two equal halves would have $k^2/4$ cut edges. Thus the vertex expansion would prefer an unbalanced clustering. To mitigate

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 410–425, 2012.

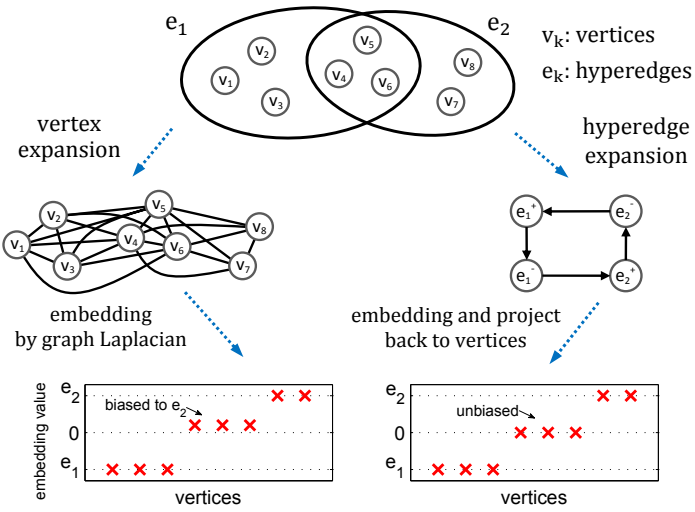


Fig. 1. An example of hypergraph embedding with two hyperedges. The hyperedge expansion embedding is unbiased, while the vertex expansion embedding (not to scale) depends on the hyperedge sizes. (see Section 4 for more details)

the problem of unbalanced clustering, it is proposed in star expansion and NHC to use the cluster volume as a normalizer for balancing the cluster sizes. But such normalization can not completely eliminate the problem. We present the following example of vertex embedding to explain why the problem still exists.

By computing the eigenvectors of the normalized Laplacian $\boldsymbol{L}_{NHC}$ of the induced graph, it is possible to project the vertices into an Euclidian space, which is called *embedding* in spectral graph learning [5]. On the left side of Figure 1, we show the 1-dimensional vertex embedding of NHC by the eigenvector corresponding to the second smallest eigenvalue of $\boldsymbol{L}_{NHC}$. It is worth to focus on the vertices that belong to both hyperedges (the overlapping part). Although the hyperedges have the same weight and the cluster volume normalizer is applied, the overlapping part is still biased to the side with less vertices (in this case e_2 side). This means that the optimal clustering of two clusters should assign the overlapping part and other vertices in e_2 to one cluster. Such bias might be a problem when the hyperedge sizes are unbalanced, e.g. co-citation relations with a lot or a few citations. Moreover, the behavior of the artificial normalization (or "correction") could be undesirable when many hyperedges intersect with each other, because the cost of the clustering would depend on how a hyperedge is split into the clusters. An even split would introduce a different cost compared to an uneven split.

As any hyperedge that is not entirely within the same cluster represents a relation that is violated by the clustering, it would be natural to have the learning result independent of the hyperedge sizes and only depend on the hyperedge connectivity and hyperedge weights. We present a new transformation called

hyperedge expansion (HE) based on a network flow technique so that the learning result is invariant to the distribution of vertices among hyperedges. HE expansion is first carried out on the hyperedge level. Then the learning results on hyperedges are projected back to the vertices through the adjacency information between hyperedges and vertices. In Figure 1, the embedding with HE expansion places the overlapping vertices in the middle without bias.

The main contributions of this paper are as follows. We formulate the HE expansion with the Laplacian of the auxiliary directed graph, which can be transformed into a quadratic eigenvalue problem that has some new properties. To our best knowledge, this is the first work to use such formulation. We also present the embedding and semi-supervised learning algorithms for hypergraphs based on HE expansion. In the experiments the proposed algorithms are compared with state-of-the-art methods and show statistically significant gains in performance.

2 Problem Statement

A hypergraph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}, w\}$ consists of a *vertex set* $\mathcal{V}$, a *hyperedge set* $\mathcal{E}$, and a weighting function $w : \mathcal{E} \to \mathbb{R}^+$. Each hyperedge $e \in \mathcal{E}$ is a subset of $\mathcal{V}$. A hyperedge e is *incident* with a vertex v if $v \in e$. The weighted degree of a vertex is $deg(v) = \sum_{v \in e, e \in \mathcal{E}} w(e)$. The degree of a hyperedge is $deg(e) = |e|$. We say a hypergraph $\mathcal{H}$ is *connected* if for any v_i, v_j there exists a hyperedge path $\{e_1, e_2, ..., e_p\}$ such that $v_i \in e_1$, $v_j \in e_p$ and $e_k \cap e_{k+1} \neq \emptyset$ $(1 \leq k < p)$. Without loss of generality, we assume the hypergraph is always connected in this paper.

The (undirected) *induced graph* $\mathcal{G}_\mathcal{H} = \{\mathcal{V}', \mathcal{E}', w'\}$ derived from the hypergraph $\mathcal{H}$ consists of the same vertex set $\mathcal{V}' = \mathcal{V}$. An edge $e' \in \mathcal{E}'$ is placed between v_i and v_j in $\mathcal{G}_\mathcal{H}$ if there exists a hyperedge e in $\mathcal{H}$ which is incident with both v_i and v_j. The edge weight is defined as $w'(e') = \sum_{e \in \mathcal{E}, e \ni v_i, v_j} w(e)/deg(e)$. One can show that the induced graph is closely related to the star expansion and NHC. In fact there is only a small difference between the Laplacian of the induced graph and the Laplacian of NHC on the main diagonal.

Let $\mathcal{Y}$ denote a set of class labels. A multi-class labeling on a hypergraph $\mathcal{H}$ is a mapping $l : \mathcal{V} \to \mathcal{Y}$ that associates each vertex v with a label $l(v)$. In this paper, only a single label is allowed for one vertex. Since l defines several clusters of vertices by the labels, we interchangeably use "clustering" or "partitioning" in the paper as "labeling". For a hyperedge e, $l(e) = \{l(v) | v \in e\}$ is the set of labels associated to e. When $|l(e)| > 1$, we say that the hyperedge e is broken or violated by l. Let $\mathcal{V}_{l,y}$ denote the set of vertices that carry label y in the labeling l, and $\mathcal{V}^c_{l,y} = \mathcal{V} \setminus \mathcal{V}_{l,y}$ denote the remaining vertices.

In a *hypergraph semi-supervised learning* (HSSL) problem, a partial labeling $\bar{l} : \bar{\mathcal{V}} \to \mathcal{Y}$ is known on a subset of vertices $\bar{\mathcal{V}} \subset \mathcal{V}$. We assume that the vertices in $\bar{\mathcal{V}}$ carry all the labels in $\mathcal{Y}$. The goal of HSSL is to find the full labeling l that coincides with $\bar{l}$ on $\bar{\mathcal{V}}$, and minimizes some objective $\min_l R(\mathcal{H}, l) \in \mathbb{R}$. We list two typical objective functions as following,

$$R_{HE} = \sum_{e \in \mathcal{E}, |l(e)|>1} w(e), \tag{1}$$

$$R_{NHC} = \sum_{y \in \mathcal{Y}} \frac{\sum_{e \in \mathcal{E}} \frac{1}{deg(e)} w(e) |e \cap \mathcal{V}_{l,y}||e \cap \mathcal{V}_{l,y}^c|}{\sum_{v \in \mathcal{V}_{l,y}} deg(v)}. \tag{2}$$

The R_{HE} is the sum of the weights of hyperedges which are broken [6], and R_{NHC} is defined in [5]. One can show that the (relaxed) optimal solution for R_{NHC} is an eigenvector of the normalized Laplacian of the induced graph.

Some existing works, for example [7] and [8], have already shown that the pairwise affinity relations after the projection to the induced graph would introduce information-loss, and working directly on the hypergraph (like the objective R_{HE}) could produce better performance. Ladicky et al. also experimentally show that the objective which is invariant to the number of objects (vertices) in each co-occurrence relation (hyperedge) [9], namely the *invariance property*, would achieve better performance on classification tasks. We can verify that R_{HE} satisfies the invariance property, while R_{NHC} does not (see Figure 1).

Although the R_{HE} has been proposed for a long time, all the existing works focus on the exact algorithms that directly optimize R_{HE} (e.g. see [10] for an early work and [11] for recent works). These combinatorial algorithms can be efficient when the number of classes is small (e.g. 2 or 3), but the exact algorithm is NP-hard for an arbitrary number of classes. On the other hand, these algorithms produce a combinatorial solution of hard clustering. There is no information about the confidence with which a vertex belongs to a cluster.

We use a different approach, i.e. the spectral technique, to target the same objective R_{HE}. The complexity of our algorithm is linear to the number of classes, and the final result is a soft clustering where the certainty of assigning a vertex to a cluster can be interpreted. To our best knowledge, this is the first work that applies the quadratic eigenvalue analysis to the R_{HE} objective.

In another task called *hypergraph embedding*, we would like to project the vertices into a low dimensional Euclidean space $\mathbb{R}^k$ where the vertices that are close to each other in the hypergraph should also stay close (as shown in Figure 1). R_{HE} and R_{NHC} actually define different notions of "closeness" in the hypergraph.

3 Hyperedge Expansion

In the simple case of two classes, in order to implement R_{HE}, we need to find a set of hyperedges of minimum weight that need to be cut to separate the vertices of the hypergraph into two parts, which is called the *minimum hyperedge cut problem* (MHCP). The optimal MHCP solution for the hypergraph shown in Figure 2 (a) would split hyperedge e_3 into two parts and the optimal R_{HE} cost is $w(e_3)$ (for the moment just consider the example hypergraph without knowing the meaning of the hyperedge subscripts).

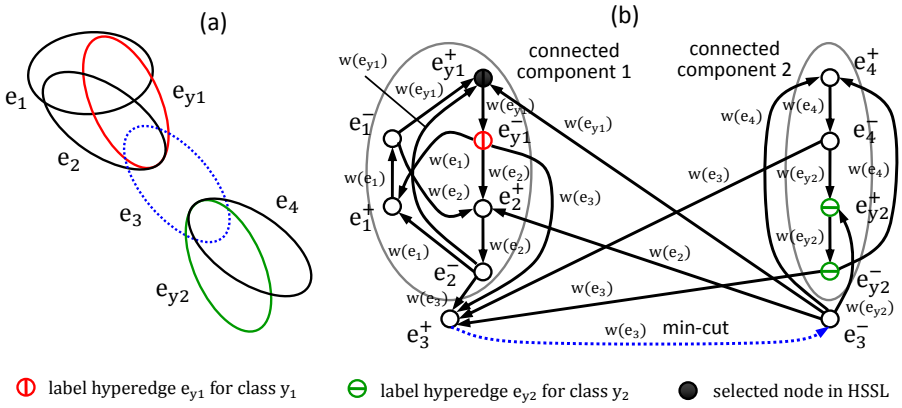


Fig. 2. (a) The original hypergraph where the hyperedge e_3 has the smallest weight. **(b)** The weighted directed graph $\hat{\mathcal{G}}$ constructed from the hypergraph. The directed edge (e_3^+, e_3^-) is the minimum cut of $\hat{\mathcal{G}}$ since its removal separates $\hat{\mathcal{G}}$ into two strongly connected components.

The technique for solving MHCP dates back to [10] where the hypergraph is transformed into a flow network and the minimum hyperedge cut is identified with a max-flow solution. We use a slightly different transformation as in [12].

The hyperedge expansion works as follows. We construct a directed graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ that includes two vertices e^+ and e^- for each hyperedge e in the original hypergraph. Note that the vertices in $\hat{\mathcal{G}}$ correspond to the hyperedges, but not the vertices in the original hypergraph. A directed edge is placed from e^+ to e^- with weight $w(e)$ where w is the weighting function in the hypergraph. For every pair of overlapping hyperedges e_1 and e_2, two directed edges (e_1^-, e_2^+) and (e_2^-, e_1^+) are added to $\hat{\mathcal{G}}$ with weights $w(e_2)$ and $w(e_1)$ (see Figure 2 (b)).

Then we can identify the solution of MHCP by finding the min-cut of $\hat{\mathcal{G}}$ that separates $\hat{\mathcal{G}}$ into at least two strongly-connected components. The correctness follows immediately from the construction of $\hat{\mathcal{G}}$. Note that the edges attached to an e^+ node have the same weights, and an e^+ node has exactly one outgoing edge. For any edge in the min-cut which goes from an e^- node to an e^+ node, we can replace it with the outgoing edge of the e^+ node and construct an equivalent min-cut. Thus the min-cut could contain only the edges from the e^+ nodes to the e^- nodes. As shown in Figure 2 (b), the min-cut solution includes only the directed edge (e_3^+, e_3^-). The cost of the min-cut is $w(e_3)$, which is exactly the same as the cost of the original MHCP.

In matrix form, the adjacency matrix of $\hat{\mathcal{G}}$ can be defined as

$$\boldsymbol{A}_{\hat{\mathcal{G}}} = \begin{bmatrix} \mathbf{0} & \boldsymbol{A}\boldsymbol{W} \\ \boldsymbol{W} & \mathbf{0} \end{bmatrix}, \tag{3}$$

where $\boldsymbol{A}$ is the $|\mathcal{E}| \times |\mathcal{E}|$ adjacency matrix of hyperedges ($\boldsymbol{A}(i,j) = 1$ if $e_i \cap e_j \neq \emptyset$ and $e_i \neq e_j$, otherwise $\boldsymbol{A}(i,j) = 0$), and $\boldsymbol{W} = \text{diag}([w(e_1), w(e_2), ...])$ is the

diagonal matrix of hyperedge weights. The index (i, j) after the matrix indicates the element at ith row and jth column. We sort the rows and columns of $\boldsymbol{A}_{\hat{\mathcal{G}}}$ in the order $[e_1^-, e_2^-, ..., e_1^+, e_2^+, ...]$, so the elements in all other matrices and vectors should follow the same order. The out-degree matrix of $\hat{\mathcal{G}}$ is $\boldsymbol{D}_{\hat{\mathcal{G}},out} = \left[\begin{smallmatrix} \boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W} \end{smallmatrix}\right]$, where $\boldsymbol{D} = \text{diag}(\boldsymbol{e}^\top \boldsymbol{W} \boldsymbol{A})$ and $\boldsymbol{e}$ is an all-ones vector. Then the out-degree Laplacian of $\hat{\mathcal{G}}$ can be defined as following

$$\boldsymbol{L} = \boldsymbol{D}_{\hat{\mathcal{G}},out} - \boldsymbol{A}_{\hat{\mathcal{G}}} = \begin{bmatrix} \boldsymbol{D} & -\boldsymbol{A}\boldsymbol{W} \\ -\boldsymbol{W} & \boldsymbol{W} \end{bmatrix}. \tag{4}$$

There are existing theories about the spectral property of the directed graph based on symmetrization of $\boldsymbol{L}$ [13,14], and the corresponding learning problem for directed graph [15]. It is shown that a Cheeger inequality can be established with the first non-trivial eigenvalue of $\tilde{\boldsymbol{L}} = \frac{\boldsymbol{V}\boldsymbol{P} + \boldsymbol{P}^\top \boldsymbol{V}}{2}$, where $\boldsymbol{P}$ is the (non-symmetric) transition probability matrix of the directed graph and $\boldsymbol{V}$ is the diagonal matrix of the first non-trivial eigenvector of $\boldsymbol{P}$. In the transition probability matrix $\boldsymbol{P}$, all out-going edge weights are normalized by the out-degree. For the special structure of $\hat{\mathcal{G}}$ in our case, the out-degree normalization could be problematic since the correct mapping from the min-cut of $\hat{\mathcal{G}}$ to the original MHCP problem relies on the special weighting of the edges. When changing the edge weights, it could be possible that the min-cut of $\hat{\mathcal{G}}$ also contains edges from the e^- nodes to the e^+ nodes, which is undesirable in our case. Instead, we avoid to use the normalized $\boldsymbol{P}$ and show that the unnormalized Laplacian $\boldsymbol{L}$ is connected to a relaxation of the min-cut problem on $\hat{\mathcal{G}}$.

Denote the nodes in the connected component on one side of the min-cut with $\mathcal{S} \subset \hat{\mathcal{V}}$, and the nodes on the other side with $\mathcal{S}^c$. We define a vector $f \in \{1/\sqrt{|\mathcal{S}|}, 0\}^{|\hat{\mathcal{V}}|}$, where $f(e)$ is the entry corresponding to the node e. $f(e) = 1/\sqrt{|\mathcal{S}|}$ if $e \in \mathcal{S}$ and otherwise 0. It can be shown that $f^\top f = 1$ and the cost of the cut can be written as

$$C = \sum_{e_1, e_2 \in \hat{\mathcal{V}}, (e_1, e_2) \in \hat{\mathcal{E}}} |\mathcal{S}| w(e_1, e_2) \left(f(e_1) - f(e_2)\right) f(e_1). \tag{5}$$

The second $f(e_1)$ ensures that only the edges from $\mathcal{S}$ to $\mathcal{S}^c$ are counted when $f(e_1) = 1/\sqrt{|\mathcal{S}|}$ and $f(e_2) = 0$. Then we relax f to take positive continuous values and find the relaxed f that minimizes C by the Lagrange multiplier method with constraint $f^\top f = 1$. When taking the partial derivatives, we drop the contributions from $f(e_2)$ which is close to zero. This implies the following approximation

$$\begin{aligned} \frac{\partial w(e_1, e_2) \left(f(e_1) - f(e_2)\right) f(e_1)}{\partial f(e_1)} &= 2w(e_1, e_2) f(e_1) - w(e_1, e_2) f(e_2) \\ &\approx 2w(e_1, e_2) f(e_1), \end{aligned} \tag{6}$$

$$\frac{\partial w(e_1, e_2) \left(f(e_1) - f(e_2)\right) f(e_1)}{\partial f(e_2)} = -w(e_1, e_2) f(e_1). \tag{7}$$

Setting the partial derivative with respect to $f(e)$ to zero, it results in a matrix form $f^\top \left(2\boldsymbol{D}_{\hat{\mathcal{G}},out} - \boldsymbol{A}_{\hat{\mathcal{G}}}\right) = 2\lambda f^\top$ where λ is the Lagrange multiplier. The matrix on the left side is the same as $\boldsymbol{L}$ except the doubled diagonal. We can also interpret 2λ as an eigenvalue and f as a left eigenvector.

For a non-Hermitian matrix like $\boldsymbol{L}$, the Courant-Fischers min-max theorem does not hold anymore. The field of values of the non-Hermitian matrix is a superset of the convex hull of the eigenvalues [16], and there is no guarantee that all the eigenvalues are real. Although $\boldsymbol{L}$ is a non-Hermitian matrix, we show in the next section that the special structure of $\hat{\mathcal{G}}$ leads to some special properties of $\boldsymbol{L}$ as addition to the properties in the general case. These special properties would allow us to carry out the learning tasks with $\boldsymbol{L}$.

4 Hypergraph Embedding

The embedding of a (hyper)graph projects the vertices into a low dimensional Euclidean space. With the NHC objective one can construct the $|\mathcal{V}| \times |\mathcal{V}|$ normalized hypergraph Laplacian $\boldsymbol{L}_{NHC} = \boldsymbol{I} - \frac{1}{2}\boldsymbol{D}_v^{-\frac{1}{2}}\boldsymbol{H}\,\boldsymbol{W}\,\boldsymbol{H}^\top\boldsymbol{D}_v^{-\frac{1}{2}}$, where $\boldsymbol{H}$ is the $|\mathcal{E}| \times |\mathcal{V}|$ incident matrix ($\boldsymbol{H}(e,v) = 1$ if $v \in e$; otherwise $\boldsymbol{H}(e,v) = 0$) and $\boldsymbol{D}_v = \text{diag}(deg(v))$ is the vertex degree matrix. Let $g_0, ..., g_{k-1}$ be the eigenvectors of $\boldsymbol{L}_{NHC}$ associated with the k smallest eigenvalues. The embedding of vertex v in a k-dimensional space is just the row vector at the v'th row of $[g_0, ..., g_{k-1}]$ [5].

We can also carry out this task with the R_{HE} objective by taking the left eigenvectors of $\boldsymbol{L}$ and mapping the hyperedge embedding back to the vertices. Suppose we have the left (real) eigenvectors $x_0, ..., x_{k-1}$ associated with the k smallest real eigenvalues of $\boldsymbol{L}$, i.e. $x_i^\top \boldsymbol{L} = \lambda_i x_i^\top,\ i \in \{0, ..., k-1\}$. Then the embedding for vertex v can be formulated as

$$\text{embedding}(v) = \left[x_0^-, ..., x_{k-1}^-\right]^\top \boldsymbol{H}(\cdot, v), \tag{8}$$

where x_i^- means the first half ($x_i(e^-)$ part) of x_i. But in the most general case the left eigenvectors could be complex for the non-Hermitian matrix $\boldsymbol{L}$. Fortunately, for most real problems, we show that all eigenvectors of $\boldsymbol{L}$ are real.

Theorem 1. *All eigenvalues of $\boldsymbol{L}$ are non-negative real numbers and the left eigenvectors of $\boldsymbol{L}$ are real if and only if there exists $\gamma \in \mathbb{R}$ such that the matrix $\boldsymbol{Q}(\gamma) = \gamma^2\,\boldsymbol{W}^{-2} + \gamma\,\boldsymbol{W}^{-1}(\boldsymbol{I} + \boldsymbol{W}^{-1}\boldsymbol{D}) + (\boldsymbol{W}^{-1}\boldsymbol{D} - \boldsymbol{A})$ is negative definite.*

Proof. Denote the eigenvalue of $\boldsymbol{L}$ by λ and the left eigenvector by $x = [x^-, x^+]$, where x^- and x^+ are the first and second halves of x. The eigenvalue problem $x^\top \boldsymbol{L} = \lambda x^\top$ can be reformulated as

$$\begin{aligned} \boldsymbol{D}x^- - \boldsymbol{W}x^+ &= \lambda x^-, \\ -\boldsymbol{W}\boldsymbol{A}x^- + \boldsymbol{W}x^+ &= \lambda x^+. \end{aligned}$$

By substituting $x^+ = \boldsymbol{W}^{-1}(\boldsymbol{D} - \lambda \boldsymbol{I})x^-$ in the second equation, we obtain a *quadratic eigenvalue problem* (QEP) $\boldsymbol{Q}(\lambda)x^- = 0$. Note that the coefficient matrices of λ^2 and λ are positive definite. It is known that a QEP is overdamped if and only if there exists $\gamma \in \mathbb{R}$ such that the matrix $\boldsymbol{Q}(\gamma)$ is negative definite and $(\boldsymbol{W}^{-1}\boldsymbol{D} - \boldsymbol{A})$ is positive semi-definite (see Theorem 2 and Definition 4 of [17]). Without loss of generality, the second condition can be always satisfied by scaling the hyperedge weights with the same factor. It is also known that the overdamped QEP $\boldsymbol{Q}(\lambda)x^- = 0$ has $2|\mathcal{E}|$ non-negative real eigenvalues, and thus $2|\mathcal{E}|$ real left eigenvectors. □

The condition stated in Theorem 1 is hard to verify in practice. The state-of-the-art techniques usually require to actually compute all the eigenvalues of the QEP. We give a sufficient condition which is easier to verify.

Corollary 1. *All eigenvalues of* $\boldsymbol{L}$ *are non-negative real numbers and the left eigenvectors of* $\boldsymbol{L}$ *are real if* $d(\boldsymbol{D}(i,i) + \boldsymbol{W}(i,i)) > 8\boldsymbol{D}(i,i)\,\boldsymbol{W}(i,i)$ *for all* $i \in \{1, ..., |\mathcal{E}|\}$, *where* $d = \min_i(\boldsymbol{D}(i,i) + \boldsymbol{W}(i,i))$.

Proof. As shown in Definition 1 of [17], the conclusion of Corollary 1 holds if

$$\left((x^-)^*\,\boldsymbol{W}^{-1}(\boldsymbol{I} + \boldsymbol{W}^{-1}\boldsymbol{D})x^-\right)^2 > 4\left((x^-)^*\,\boldsymbol{W}^{-2}x^-\right)\left((x^-)^*(\,\boldsymbol{W}^{-1}\boldsymbol{D} - \boldsymbol{A})x^-\right)$$

for all non-zero $x^- \in \mathbb{C}^{|\mathcal{E}|}$, where $(x^-)^*$ denotes the conjugate transpose of x^-. Let $z = \boldsymbol{W}^{-1}x^-$ and note that $\boldsymbol{W}^{-1}$ is a diagonal matrix with positive main diagonal. We can transform the above condition into

$$\left(\frac{z^*(\,\boldsymbol{W} + \boldsymbol{D})z}{z^*z}\right)^2 > \frac{z^*\,\boldsymbol{W}(4\,\boldsymbol{W}^{-1}\boldsymbol{D} - 4\boldsymbol{A})\,\boldsymbol{W}z}{z^*z}$$

for all non-zero $z \in \mathbb{C}^{|\mathcal{E}|}$. Both sides of the inequality contain a Rayleigh quotient. It can be shown that $d = \min_z \frac{z^*(\boldsymbol{W}+\boldsymbol{D})z}{z^*z} = \min_i(\boldsymbol{D}(i,i) + \boldsymbol{W}(i,i)) > 0$. Therefore a sufficient condition is

$$\frac{z^*\,(d(\,\boldsymbol{W} + \boldsymbol{D}) - 4\boldsymbol{D}\,\boldsymbol{W} + 4\,\boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{W})\,z}{z^*z} > 0$$

for all non-zero $z \in \mathbb{C}^{|\mathcal{E}|}$, which means that the Hermitian matrix $\boldsymbol{R} = (d(\,\boldsymbol{W} + \boldsymbol{D})\, -4\boldsymbol{D}\,\boldsymbol{W} + 4\,\boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{W})$ must be positive definite. We know that $\boldsymbol{R}$ is positive definite if $\boldsymbol{R}$ is strictly diagonally dominant and has all positive diagonal entries. Noting that each row of $(\,\boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{W} - \boldsymbol{D}\,\boldsymbol{W})$ sums up to 0, we obtain the sufficient condition in Corollary 1. □

In fact we find that all the hypergraphs tested in the experimental section satisfy this sufficient condition except the dataset *AmazonBook*. But the first 6 eigenvalues (smallest magnitude) of the hypergraph constructed from *AmazonBook* are all real non-negative numbers. Experiments in Section 6 show that the hyperedge expansion embedding works well in general.

5 Hypergraph Semi-supervised Learning

Like the existing works (e.g. NHC and [18]), we convert the multi-class HSSL problem into a set of binary classification problems. We pick up one class y each time, compute a *class score* for each unlabeled vertex v that indicates the possibility of v belonging to class y, and repeat this procedure for all labels. Finally the label with the highest class score is assigned to v.

5.1 Computing Class Scores

The desired procedure should take the hypergraph $\mathcal{H}$, the partial labeling $\bar{l}$, and the chosen class y as input, while output the class scores for all unlabeled vertices. The class score $\text{score}(v, y)$ can be defined as the reciprocal of the "distances" from the labeled vertices of label y to an unlabeled vertex v. An intuitive $\text{score}(v, y)$ could be the reciprocal of the average commute distance in the induced graph from v to each vertex of label y, which can be computed by the generalized inverse of $\boldsymbol{L}_{NHC}$ [19]. Or $\text{score}(v, y)$ could be obtained by simulating a random walk on the induced graph with restart from labeled vertices of label y [20]. Here we compute the scores based on $\boldsymbol{L}$ with hyperedge expansion.

To incorporate the partial labeling, some auxiliary hyperedges have to be added to the hypergraph. For each label $y \in \mathcal{Y}$, we create a *label hyperedge* containing all the vertices in $\bar{\mathcal{V}}$ with label y. In other words, a new hyperedge $e_y = \bar{l}^{-1}(y)$ is added to the original hypergraph, which is illustrated in Figure 2 (a) as e_{y1} and e_{y2}. The weights of all label hyperedges are set to a pre-defined value w_l, i.e. $w(e_{y1}) = w(e_{y2}) = ... = w_l$.

In each step one class is selected (e.g. the chosen class is $y = y_1$ in Figure 2). Then we define the class modified Laplacian

$$\boldsymbol{L}_y = \boldsymbol{L} - \alpha \boldsymbol{B}, \quad \boldsymbol{B} = \begin{bmatrix} \ddots & & \\ & w(e_y) & \\ & & \ddots \end{bmatrix} \tag{9}$$

where α is a parameter and $\boldsymbol{B}$ has only one non-zero entry $w(e_y)$ in the bottom-right half diagonal corresponding to the position of e_y^+. A larger α would have a bigger influence on guiding the direction of the hyperedge partition around e_y^+, while a smaller α would let the partition follow the intrinsic principle direction of $\hat{\mathcal{G}}$. Note that for each class $y \in \mathcal{Y}$ the matrix $\boldsymbol{L}_y$ has to be recomputed. We denote the left eigenvector of $\boldsymbol{L}_y$ by f_y.

With the commonly-used symmetric Laplacian, the eigenvector associated with the second smallest eigenvalue, namely the *Fiedler vector*, is often taken to partition the graph. For $\boldsymbol{L}_y$ with arbitrary $\alpha > 0$, we show that the eigenvector of $\boldsymbol{L}_y$ with the smallest real eigenvalue has the following property.

Theorem 2. *When $\alpha > 0$, there exists one eigenvalue λ_y^0 of $\boldsymbol{L}_y$ which is real and has the smallest real part among all eigenvalues of $\boldsymbol{L}_y$. The left eigenvector f_y^0 corresponding to λ_y^0 has all positive entries. Furthermore we have $\lambda_y^0 \geq -\alpha w(e_y)$.*

Proof. Consider the matrix $\boldsymbol{L}'_y = \mu \boldsymbol{I} - \boldsymbol{L}_y$ where $\boldsymbol{I}$ is the identity matrix and $\mu > 0$. Since the underlying graph is strongly connected, the matrix $\boldsymbol{L}'_y$ is non-negative and irreducible for some μ. By the Perron-Frobenius theorem, there exists an all positive left eigenvector f_y^0 and an eigenvalue $\mu - \lambda_y^0$, which is real and has the biggest magnitude. Thus f_y^0 is a left eigenvector of $\boldsymbol{L}_y$ corresponding to λ_y^0. The bound of λ_y^0 directly follows the spectral radius bound of the Perron-Frobenius theorem. □

Then we can compute the score of each unlabeled vertex v as the sum of all $f_y^0(e^-)$ values where e is a hyperedge and $e \ni v$, i.e. $\text{score}(v, y) = \sum_{e \ni v} f_y^0(e^-)$. This score is repeatedly computed for each class $y \in \mathcal{Y}$ for v, and finally v is assigned to the class with the highest score. We could analogically justify the usage of f_y^0 as in the argument in the end of Section 3. The only difference is that all the vertices labeled with y should be assigned to the same side of the min-cut. This can be modeled as a soft constraint with the term $\alpha \boldsymbol{B}$ in (9).

5.2 The Algorithm and Complexity

Summing up all the procedures above, we obtain the complete HE expansion algorithm for HSSL. Only two parameters are required for the algorithm: the weight for label hyperedges w_l and the parameter α. Empirically it is a good choice to set w_l to the largest weight of all hyperedges.

Algorithm 1. : $l = \textbf{HSSL-HE}(\mathcal{H} = \{\mathcal{V}, \mathcal{E}, w\}, \bar{l} : \bar{\mathcal{V}} \to \mathcal{Y}, w_l, \alpha)$

Let $\mathcal{E}_{ex} = \mathcal{E} \cup \{e_y | y \in \mathcal{Y}\}$, where $e_y = \bar{l}^{-1}(y)$ of weight w_l
Compute $\boldsymbol{L}$ from $\mathcal{H} = \{\mathcal{V}, \mathcal{E}_{ex}, w_{ex}\}$ (see (4))
Initialize the score matrix $\boldsymbol{S}$ of size $|\mathcal{E}_{ex}| \times |\mathcal{Y}|$
for all $y \in \mathcal{Y}$ **do**
    Compute the matrix $\boldsymbol{L}_y = \boldsymbol{L} - \alpha \boldsymbol{B}$ (see (9))
    Compute the left eigenvector f_y^0 of $\boldsymbol{L}_y$ corresponding to the smallest real eigenvalue
    Fill in the y's column of $\boldsymbol{S}$ with the $f_y^0(e^-)$ part, i.e., the first half of f_y^0
end for
for all $v \in \mathcal{V} \setminus \bar{\mathcal{V}}$ **do**
    Let $l(v) = \arg\max_{y \in \mathcal{Y}} \sum_{e \ni v, e \in \mathcal{E}_{ex}} \boldsymbol{S}(e, y)$
end for
return $l = l \cup \bar{l}$

The HSSL-HE algorithm involves the computation of the eigenvector of a matrix of size $2N \times 2N$ ($N = |\mathcal{E}| + |\mathcal{Y}|$), and this procedure has to be repeated $|\mathcal{Y}|$ times. It is known that each eigenvalue problem can be solved in time $O(nN^2)$ by power iteration methods like Lanczos algorithm, where n is the number of iterations. The lower bound in Theorem 2 can be used as a good initial guess of the eigenvalue. If the connectivity between hyperedges is sparse, we can further reduce the time of computing eigenvector to $O(nN)$ and the total time complexity would be $O(nN|\mathcal{Y}|)$. Generally, HSSL-HE would have better scalability when

the number of instances (vertices) is very large and the number of co-occurrence relations (hyperedges) is relatively small. Such scenarios can be found in many real applications like categorical data and census data. On the other hand, the spectral methods that operate on the induced graphs, e.g. star expansion and NHC, need $O(n|\mathcal{E}'|)$ time where $|\mathcal{E}'|$ is the number of edges created in the induced graph.

6 Experimental Results

In this section, we present results on two tasks. First, we test the proposed semi-supervised algorithm on datasets from different domains and compare the performances with the state-of-the-art methods. Second, we present the result of the HE expansion embedding.

6.1 Experiment Settings

All the classification tasks are conducted in a transductive manner: we first create a hypergraph from raw data. Then the hypergraph and some (small amount of) vertex labels are taken as inputs and the algorithm predicts the labels of the unlabeled vertices. When evaluating the algorithms in repeated runs, the labeled vertices are randomly chosen from the vertex set such that every class has at least one labeled vertex, but the same set of labeled vertices is applied to all tested algorithms in each run. For evaluation we mainly use the macro-averaged F-score.

Algorithms for Comparison: since our proposed algorithm belongs to the family that only uses relational information, we choose four state-of-the-art relational-only approaches and one feature-based approach (AnchorGraph) for comparison [1]:

(1) the hMETIS toolkit [6] is a commonly used tool for hypergraph partitioning, which optimizes the R_{HE} objective with a heuristic algorithm. Although hMETIS is mainly designed for VLSI applications, reports show that this toolkit can be applied to general classification/clustering problems [21]. We use the "preassignment of vertices" input file with hMETIS to assign the known labels in the semi-supervised task.

(2) the *normalized hypergraph cut* (NHC) algorithm by Zhou et al. [5] first transforms the hypergraph into an induced graph whose edge weights are normalized by the hyperedge sizes. Then NHC adopts the normalized Laplacian $\boldsymbol{L}_{NHC}$ to the semi-supervised setting. The NHC algorithm is the most representative approach among those based on vertex expansions.

(3) the *rendezvous algorithm* (Rend.) [22] is a semi-supervised learning approach based on a random walk on a graph. The algorithm first constructs a directed graph from the k-nearest neighbors in which all the labeled vertices

[1] The implementation of our proposed algorithm (HSSL-HE) can be found in `http://lia.epfl.ch/index.php/research/relational-learning`

have only incoming edges and thus act as absorbing states of the random walk. Then the algorithm simulates a set of particles that start a random walk from each unlabeled vertex and stop at some labeled vertices. Intuitively, a particle from an unlabeled vertex will stop at a labeled vertex of its true label with higher probability. The algorithm determines the labels based on the outcome of the random walk. We use the distances in the induced graph (the same as NHC) to construct the directed k-NN graph. We also apply a Gaussian kernel function to the distances as instructed by the author.

(4) the *semi-supervised kernel k-means* (SSKKmeans) [23] is an extension of the kernel k-means method where the kernel function is a linear combination of the graph kernel and the label-induced modifier. The label-induced component includes both same-class rewards and different-classes penalties. Again we use the induced graph from the hypergraph (the same as NHC) to compute the graph kernel.

(5) the AnchorGraph algorithm [24] focuses on the scalability of semi-supervised learning. Instead of constructing a k-NN graph from the original data, AnchorGraph chooses a small set of anchors which connect to the s-nearest neighbors in the original data, and represents each data point with a linear combination of the anchors. The semi-supervised algorithm is faster because the values to learn are only the weights of the anchors rather than the labels of the original data.

In the experiments, we use 13 relational-only datasets from three different domains to evaluate the above algorithms. The *AmazonBook co-purchase dataset* contains the books in Amazon.com and the list of books that are co-purchased [25]. We take three subsets of book products to construct the hypergraphs, where a vertex represents a book, and a hyperedge represents a co-purchase list of books. The label of each vertex is simply the category of the corresponding book. The parameter α for algorithm 1 is set to 1 for *AmazonBook*. We also construct co-citation hypergraphs from the commonly-used *Cora*, *citeseer*, and *WebKB* data. For *Cora* and *citeseer*, a vertex represents a paper, and a hyperedge contains all the papers that cite the same paper. For *WebKB* data (*cornell* and *texas*), besides the link information, word-based content information is also available. So we create some additional hyperedges that include all the papers or webpages that contain the same word. In order to show how the link information could help in classification, the hypergraphs using only contents (denoted by C) and contents plus links (denoted by CL) are respectively constructed for each *WebKB* dataset. The parameter α is set to 50 for co-citation datasets. In the last domain, categorical dataset, every instance has a set of nominal attributes which could take values from a finite set. We use 4 labeled categorical datasets *zoo*, *letter*, *20newsgroups*, and *covertype* from the UCI repository. For each dataset, a hypergraph is constructed by taking instances as vertices and creating a hyperedge for each value of the attributes. Then every hyperedge contains the instances that share the same attribute value. We discretize those attributes whose value is an integer with a range larger than 10 into 10 sections of the same size. Some tested algorithms (SSKKmeans, NHC, and Rendezvous) do not scale well on *letter*, *20newsgroups*, and *covertype*, so only a subset is tested for

Table 1. The averaged macro F-scores (and the standard deviation in the parentheses) on 13 datasets. The algorithms are tested on *AmazonBook* (*AB*) and *covertype* with 10 runs, co-citation data with 50 runs, other categorical data with 100 runs. Some information about the dataset is shown below the dataset name (#labeled vertices / #all vertices / #classes). The bold number indicates a algorithm that performs significantly better than others (p-value < 0.05 in paired t-test). The Rendezvous algorithm cannot return a result in a reasonable time period for *AmazonBook*, *Cora* and *citeseer*.

dataset	hMETIS	SSKKmeans	AnchorGraph	Rend.	NHC	HE
AB3 (100/24500/3)	0.565(0.022)	0.446(0.015)	0.519(0.025)	—	0.645(0.019)	**0.657**(0.023)
AB4 (100/18120/4)	0.517(0.107)	0.376(0.016)	0.561(0.058)	—	0.765(0.046)	**0.798**(0.023)
AB5 (80/6965/5)	0.525(0.040)	0.357(0.067)	0.472(0.046)	—	0.724(0.087)	0.716(0.064)
Cora (40/1961/7)	0.477(0.054)	0.449(0.048)	0.500(0.050)	—	0.613(0.046)	**0.637**(0.040)
citeseer (40/1318/6)	0.492(0.046)	0.361(0.030)	0.401(0.038)	—	**0.518**(0.046)	0.509(0.046)
cornell-CL (20/195/5)	0.275(0.055)	0.411(0.091)	0.417(0.058)	0.304(0.068)	0.320(0.091)	**0.497**(0.047)
cornell-C (20/195/5)	0.279(0.057)	0.427(0.092)	0.425(0.059)	0.299(0.056)	0.346(0.069)	**0.480**(0.050)
texas-CL (20/187/5)	0.238(0.028)	0.362(0.050)	0.317(0.066)	0.249(0.042)	0.268(0.089)	**0.425**(0.045)
texas-C (20/187/5)	0.236(0.039)	0.350(0.047)	0.338(0.050)	0.254(0.047)	0.267(0.098)	**0.410**(0.068)
zoo (15/100/7)	0.467(0.066)	0.822(0.058)	0.803(0.075)	0.571(0.088)	0.359(0.147)	**0.832**(0.052)
letterAE (50/1022/5)	0.379(0.049)	0.629(0.023)	**0.664**(0.039)	0.543(0.039)	0.606(0.047)	0.627(0.028)
20newsgroups (50/1067/4)	0.489(0.080)	0.480(0.041)	0.552(0.042)	0.482(0.069)	**0.642**(0.033)	0.628(0.042)
covertype (50/6344/7)	0.164(0.017)	0.285(0.019)	0.238(0.016)	0.268(0.022)	0.254(0.064)	**0.307**(0.028)

each of them. We set $\alpha = 1$ for *20newsgroups* and $\alpha = 100$ for other categorical datasets. Weighting the hyperedges usually depends on the domain knowledge. For simplicity, we assign the same weights to all hyperedges in the experiments.

6.2 Main Results

As shown in Table 1, HE performs significantly better than other methods in most cases. For some datasets, hMETIS does not work very well, partially because it is mainly designed for VLSI applications, but not general classification tasks. For *cornell* and *texas*, we can observe an improvement from C to CL with the HE algorithm, which confirms that the link information does help in classifying webpages. This improvement, however, does not exist with other algorithms.

Nevertheless, the algorithms directly designed for hypergraphs (hMETIS, NHC and HE) generally perform significantly better than those based on graphs

(SSKKmeans) or feature vectors (AnchorGraph). It suggests that hypergraph approaches would be better choices when the data is naturally organized as co-occurrence relations. For *letterAE*, the AnchorGraph actually works best, mainly because the original attributes of *letterAE* are all integer values (such as the mean of x-position of the pixels) rather than nominal variables. When the data naturally follows some pattern in a continuous metric space, methods like AnchorGraph could be better for capturing the underlying regularity.

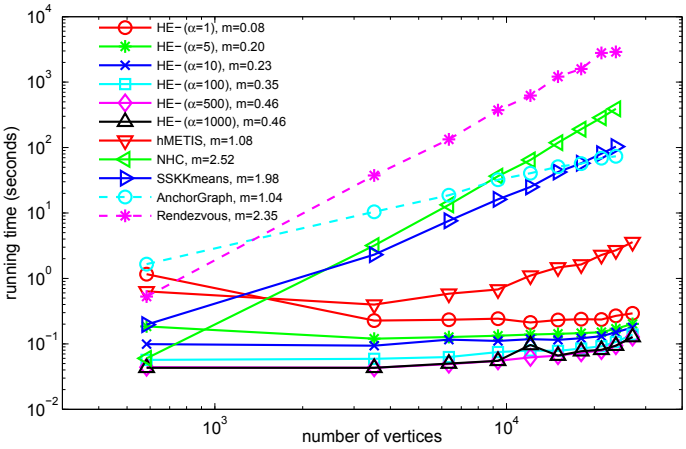


Fig. 3. The measured running times of different algorithms with 100 labeled vertices on subsets of *covertype*. The slope m of each curve is shown in the legend, which is computed by the least square fitting.

The running time of different algorithms is tested with subsets of *covertype* whose vertex set sizes range from 583 to 27056. These subsets are randomly extracted from the original data. Figure 3 shows the measured times in log scale. We have shown that the complexity of the HE algorithm mainly depends on the size of hyperedge set rather than the vertex set. Generally, the HE algorithm always stays in the same running time level regardless of the vertex set size, because the number of hyperedges in each subset does not change too much (from 122 to 143). Therefore the HE algorithm can be orders of magnitude faster than the approaches based on the induced graph when the number of hyperedges is smaller than the number of vertices. By increasing the parameter α, we can observe that HE runs faster due to the higher convergence rate of the eigenvector computation. In practice, the choice of α also depends on the classification performance, but the running time would not change by more than an order of magnitude when tuning α. The running times of hMETIS and AnchorGraph approximately grow linearly with respect to the number of vertices, while for NHC, SSKKmeans and Rendezvous the running time grows quadratically.

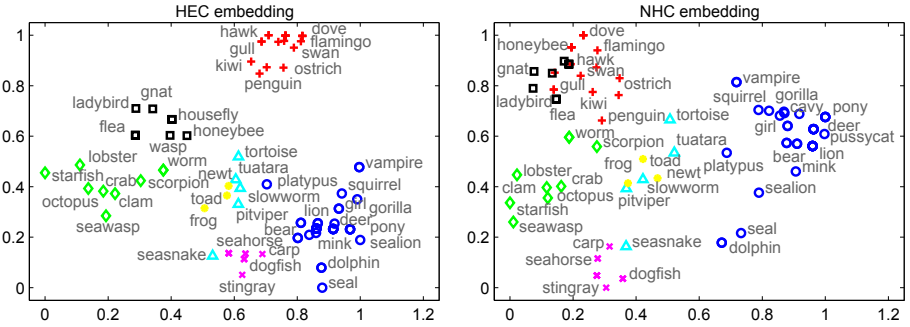


Fig. 4. The vertex embeddings of *zoo* with the eigenvectors (scaled) corresponding to the 2nd and 3rd smallest eigenvalues (for both HE and NHC embedding)

We use both HE embedding and NHC embedding to project the vertices (animals) of *zoo* into a low dimensional space. The results are shown in Figure 4. It can be seen that the HE embedding generates a different picture compared to the NHC embedding. In general the HE embedding shows a clearer separation between different classes in the 2-dimensional space, but for some instances (e.g. `seasnake` and `platypus`) both embeddings fail to give them a clear affiliation, mainly due to their special attributes.

7 Conclusion and Future Work

In this paper we propose a new formulation called hyperedge expansion and new algorithms for the semi-supervised learning and embedding tasks of hypergraph. Compared to the existing methods, the learning results with the hyperedge expansion is less sensitive to the hyperedges sizes when the data is organized with co-occurrence relations.

Our preliminary work has shown that the hyperedge expansion would be generally better than the vertex expansions when the average Jaccard coefficient between the hyperedges is high. Thus it is interesting to theoretically further investigate the applicable scopes of vertex expansions and hyperedge expansion. Moreover, we are interested in applying the hyperedge expansion technique to a broader range of real problems such as social networks and biological networks.

References

1. Chung, F.: The Laplacian of a hypergraph. Expanding graphs (DIMACS series), pp. 21–36 (1993)
2. Storm, C.: The zeta function of a hypergraph. The Electronic Journal of Combinatorics 13(R84) (2006)
3. Balof, B., Storm, C.: Constructing isospectral non-isomorphic digraphs from hypergraphs. Journal of Graph Theory 63(3), 231–242 (2010)

4. Agarwal, S., Branson, K., Belongie, S.: Higher order learning with graphs. In: Proceedings of the 23rd ICML (2006)
5. Zhou, D., Huang, J., Scholkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. In: Advances in Neural Information Processing Systems (2007)
6. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: application in VLSI domain. In: Proceedings of the 34th Annual Design Automation Conference (1997)
7. Shashua, A., Zass, R., Hazan, T.: Multi-way Clustering Using Super-Symmetric Non-negative Tensor Factorization. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part IV. LNCS, vol. 3954, pp. 595–608. Springer, Heidelberg (2006)
8. Bulò, S., Pelillo, M.: A game-theoretic approach to hypergraph clustering. In: Advances in Neural Information Processing Systems (2009)
9. Ladicky, L., Russell, C., Kohli, P., Torr, P.H.S.: Graph Cut Based Inference with Co-occurrence Statistics. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 239–253. Springer, Heidelberg (2010)
10. Lawler, E.: Cutsets and partitions of hypergraphs. Networks 3(3), 275–285 (1973)
11. Fukunaga, T.: Computing Minimum Multiway Cuts in Hypergraphs from Hypertree Packings. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 15–28. Springer, Heidelberg (2010)
12. Acid, S., Campos, L.: An algorithm for finding minimum d-separating sets in belief networks. In: Proceedings of the 12th UAI (1996)
13. Wu, C.: On Rayleigh-Ritz ratios of a generalized Laplacian matrix of directed graphs. Linear Algebra and Its Applications 402, 207–227 (2005)
14. Chung, F.: Laplacians and the Cheeger inequality for directed graphs. Annals of Combinatorics 9(1), 1–19 (2005)
15. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: Proceedings of the 22nd ICML (2005)
16. Horn, R., Johnson, C.: Topics in Matrix Analysis. Cambridge University Press (1991)
17. Guo, C., Lancaster, P.: Algorithms for hyperbolic quadratic eigenvalue problems. Mathematics of Computation, 1777–1791 (2005)
18. Sun, L., Ji, S., Ye, J.: Hypergraph spectral learning for multi-label classification. In: Proceeding of the 14th ACM SIGKDD (2008)
19. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007)
20. Lin, F., Cohen, W.: Semi-supervised classification of network data using very few labels. In: International Conference on Advances in Social Networks Analysis and Mining, pp. 192–199 (2010)
21. Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617 (2003)
22. Azran, A.: The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In: Proceedings of the 24th ICML (2007)
23. Kulis, B., Basu, S., Dhillon, I., Mooney, R.: Semi-supervised graph clustering: a kernel approach. In: Proceedings of the 22nd ICML (2005)
24. Liu, W., He, J., Chang, S.: Large graph construction for scalable semi-supervised learning. In: Proceedings of the 27th ICML (2010)
25. SNAP, http://snap.stanford.edu/data/amazon-meta.html

Nearly Exact Mining of Frequent Trees in Large Networks

Ashraf M. Kibriya and Jan Ramon

Department of Computer Science
Katholieke Universiteit Leuven
Leuven, Belgium
{ashraf.kibriya,jan.ramon}@cs.kuleuven.be

Abstract. Mining frequent patterns in a single network (graph) poses a number of challenges. Already only to match one path pattern to a network (upto subgraph isomorphism) is NP-complete. Matching algorithms that exist, become intractable even for reasonably small patterns, on networks which are large or have a high average degree. Based on recent advances in parameterized complexity theory, we propose a novel miner for rooted trees in networks. The miner, for a fixed parameter k (maximal pattern size), can mine all rooted trees with *delay* linear in the size of the network and only mildly exponential in the fixed parameter k (2^k). This allows us to mine tractably, rooted trees, in large networks such as the WWW or social networks. We establish the practical applicability of our miner, by presenting an experimental evaluation on both synthetic and real-world data.

1 Introduction

Mining frequent patterns is one of the fundamental tasks of data mining. Traditionally, patterns have consisted of simple sets of items. However, since the last decade interest has been building up in mining more structured forms of patterns, such as trees and arbitrary graphs. This has especially been the case due to the phenomenal growth of structured data sources such as the WWW, and social and citation networks.

There are generally two settings for mining in graphs. A first graph mining setting is the transactional setting, where we are given a set of graphs and a threshold t, and we want to find patterns that occur in at least t graphs in the set. The second graph mining setting, which we will consider in this paper, is the single network setting, where we are given a single graph and a threshold t, and we want to find patterns that have a support of at least t in the given single graph according to some appropriate frequency measure.

Central to any pattern mining task is the notion of pattern matching. In graph mining, subgraph isomorphism is usually the matching operator of choice. Checking for even a simple path in a graph under subgraph isomorphism is well known to be NP-complete. Indeed, the problem of finding a path in a graph can be reduced to the Hamiltonian path problem. However, recent advances in the

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 426–441, 2012.

theory of parameterized complexity have produced algorithms that can tractably solve several of the computationally hard graph problems, including subgraph isomorphism, when certain parameters of the problems are bounded. The work of [20] is particularly relevant in this case. It gives a randomized algorithm for deciding subgraph isomorphism of a tree in a network with an asymptotic complexity of $O(k^2 \log^2(k) m 2^k)$, where m is the number of edges in the network and k the size of the pattern.

In this paper we build on the work of [20]. We present an algorithm to mine *all* frequent rooted tree patterns with delay $O(k^2 \log^2(k) m 2^k)$, i.e. the time between any two consecutive frequent patterns being output is bounded by $O(k^2 \log^2(k) m 2^k)$. We present an implementation and experiments on both synthetic and real-world data. To the best of our knowledge, our work is the first to tractably mine tree patterns under subgraph isomorphism in single networks.

The rest of the paper is structured as follows. In the next section we give a brief overview of related work, followed by a section of preliminaries required for explaining our work. We then proceed to presenting our work by building on the work of [20], followed by a thorough experimental evaluation to establish the practical applicability of our mining algorithm. We then conclude with some final remarks, and future direction of our research.

2 Related Work

Most of the work done so far in graph mining is in the transactional setting. For example, graph miners AGM [18], FSG [21], FFSM [16], gSpan [30], MoFa/MoSS [3,4] and Gaston [25], are all designed for the transactional setting. These miners employ a variety of optimizations to reduce the overall runtime and memory use, including canonical forms to avoid duplicate generation of candidates and extraneous isomorphism tests. The earlier generation of miners, AGM and FSG, work in an apriori style fashion, whereas the newer generation (FFSM and beyond) use depth first search (instead of apriori style breadth-first) to further optimize memory use [29]. Furthermore, as in itemset mining, considerable work has also been done in pattern summarization, by mining more representative and comprehensive graph patterns such as closed graphs [31,7] and maximal graphs [17,27].

For mining in single networks, work has so far been limited. To our knowledge, the only ones that exist are for mining evolution of networks [1,2], node/edge classification [15,12], or mining that uses homomorpshism as matching operator [11]. Even though in [1] the authors mine patterns when mining their evolution rules, the language of the patterns is specific to time evolving networks, and excludes more general (unlabeled) patterns like trees or cycles.

For pattern matching, with (sub)graph isomorphism as the matching operator, usually the miners use one of a number of base matching algorithms. For general arbitrary graphs, Ullman[28] and VF2[8] are often popular choices, whereas Nauty[22] is also sometimes used if the matching operator is restricted graph isomorphism. Ullman and VF2 are both branch-and-bound based algorithms

that employ backtracking and pruning strategies to eliminate large parts of the search space. Nauty on the other hand, uses results from group theory to create unique canonical labelings for (automorphic) graphs that are then compared for equivalency. Note that graph isomorphism is just a special case of subgraph isomorphism, and any method for subgraph isomorphism can generally also be used for graph isomorphism.

VF2 is newer and in comparison provides mostly better runtime than Ullman, whereas in comparison to Nauty (on graph isomorphism) it is usually better on real-world structured graphs [9].

An alternative to subgraph isomorphism is to use homomorphism. Homomorphism, as used in [11], has a lower computational cost (only polynomial in the pattern size), but has other disadvantages. One of the major reasons why interest in mining conjunctive queries has been declining is that candidate generation is problematic, as illustrated by the fact that no optimal refinement operator exists [23,24]. Moreover, depending on the application requiring pattern vertices to map to different network vertices (as in isomorphism) may be the most natural choice.

3 Preliminaries

3.1 Graphs

We recall basic graph theoretic notions used in this paper. For more background in this area, see also [10]. A *graph* is a pair $G = (V_G, E_G)$, where $V_G \neq \emptyset$ is a finite set of vertices/nodes, and $E_G \subseteq \{\{x, y\} \mid x, y \in V_G\}$ a set of edges connecting those vertices. For any graph G its vertex and edge set will also be referred to as $V(G)$ and $E(G)$ respectively. If $\{u, v\} \in E(G)$, we say u and v are *adjacent* vertices and the edge $\{u, v\}$ is *incident* with the vertex v. In this paper, we call $|V(G)|$ the *size* of G. A *path* in a graph G is a sequence $\{v_1, v_2, \ldots, v_n\}$ of pairwise distinct vertices of G such that $\{v_i, v_{i+1}\} \in E(G)$ for all $1 \leq i < n$. A *tree* is a graph such that there is a unique path between any pair of its vertices. A rooted tree is a tree T in which a single vertex $r \in V(T)$, denoted by $root(T)$, is distinguished and is called the root.

A *labeled graph* is a quadruple $G = (V_G, E_G, \Sigma_G, \lambda_G)$, where (V_G, E_G) is a graph, $\Sigma_G \neq \emptyset$ a set of labels, and $\lambda_G : V_G \cup E_G \rightarrow \Sigma_G$ a function assigning labels to vertices and edges.

A graph $H = (V_H, E_H)$ is called a *subgraph* of $G = (V_G, E_G)$, if $V_H \subseteq V_G$ and $E_H \subseteq E_G$. It is an *induced* subgraph if for all $\forall v \in V_H, (v, v') \in E_H \Leftrightarrow (v, v') \in E_G$, otherwise it is a *non-induced* subgraph.

A graph $H = (V_H, E_H)$ is said to be *isomorphic* to $G = (V_G, E_G)$ (denoted by $H \cong G$), if there exists an edge-preserving bijective mapping of H onto G. For labeled graphs the mapping, in addition to edge-preserving, also has to be label-preserving. Formally, $H \cong G$ if there exists a function $\varphi : V_H \rightarrow V_G$ such that $\forall u, v \in V_H, (u, v) \in E_H \Leftrightarrow (\varphi(u), \varphi(v)) \in E_G$, and for the labeled case additionally, $\forall u \in V_H, \lambda_H(u) = \lambda_G(\varphi(u))$. If H is isomorphic to a subgraph of

G, then we call H *subgraph isomorphic* to G and write $H \preceq G$. In that case, the mapping is called an *embedding* of H in G.

If a mapping from H to G is edge-preserving but not bijective (and hence not one-to-one), then it defines a *homomorphism* between H and G.

We denote with $\text{Emb}(H, G)$, the set of all isomorphic embeddings of H in G. Note, that (i) the number of embeddings $|\text{Emb}(H, G)|$ can be exponential, and (ii) that in this paper we consider normal subgraph isomorphism rather than the more restrictive induced subgraph isomorphism.

3.2 Group Theory

A *group* G is a set of elements endowed with an arbitrary binary operation $(\cdot)$, such that it satisfies the following four properties, known as the group axioms:

- **Closure.** If $a, b \in G$, then so does $a \cdot b$.
- **Associativity.** $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- **Identity.** There exists an element $i \in G$, such that for every $a \in G$: $a \cdot i = i \cdot a = a$.
- **Inverse.** For every $a \in G$, there exists an element a^{-1}, such that $a \cdot a^{-1} = i$

A *ring* is a set of elements with two specified binary operations, addition $(+)$ and multiplication $(\times)$. It must satisfy all the group axioms for $(+)$, all but the inverse axiom for $(\times)$, and the following additional axioms:

- **Commutativity of** $(+)$**.** For all $a, b \in G$, $a + b = b + a$.
- **Distributivity of** $(\times)$ **over** $(+)$**.** For all $a, b, c \in G$, $a \times (b+c) = a \times b + a \times c$ and $(a + b) \times c = a \times c + b \times c$.

If in addition to the above if it also satisfies commutativity of $(\times)$, then it is called a *commutative ring*, otherwise a *non-commutative ring*.

A *field* is a commutative ring for which also the inverse axiom for $\times$ holds.

For a ring R, an integer n and $a \in R$, $n.a = \sum_{i=1}^{n} a$ is called the scalar multiplication between n and a. For any finite field F, it is necessarily the case that there exists an integer $n > 0$, such that for every $a \in F$, $na = 0$. The smallest such n for a field is called its *characteristic*.

4 Problem Statement

Let $\mathcal{G}$ be the *language* of all graphs, let $\mathcal{L}_{\mathcal{P}} \subseteq \mathcal{G}$ be a language of *patterns*, let $M(\mathcal{L}_{\mathcal{P}}, \mathcal{G})$ be some measure of interestingness, let t be a given threshold of interestingness and $G \in \mathcal{G}$ be the network we want to mine patterns in. Then, we would like to compute the set $\mathcal{F}_{(\mathcal{L}_{\mathcal{P}}, \mathcal{G})}$ of interesting patterns defined by:

$$\mathcal{F}_{(\mathcal{L}_{\mathcal{P}}, \mathcal{G})} = \{T \in \mathcal{L}_{\mathcal{P}} : M(T, G) \geq t\}$$

In our case, our pattern language is the class of rooted trees.

Several frequency measures have been proposed, as measures of interestingness, in the literature for single graph mining [5,6]. This paper primarily focuses

on the matching of patterns, and though our methods are general, for simplicity we restrict ourselves to the frequency measure obtained by counting the number of possible images of the root of a rooted tree pattern. This support measure is defined as follows.

Definition 1 (root image). *The root image of a rooted tree T in G is the set of all vertices $v \in G$ to which $root(T)$ can be mapped under subgraph isomorphism, i.e.,*

$$\mathcal{RI}(T, G) = \{\varphi(root(T)) \mid \varphi \in Emb(T, G)\},$$

Definition 2 (support). *Let T be a rooted tree and G be a graph. Then, we define the support of T in G as the size of its root image, i.e.,*

$$supp(T, G) = |\mathcal{RI}(T, G)|.$$

This support measure is anti-monotone w.r.t. increasing pattern size.

For the remainder of the paper, we will consider a network G and for brevity we will use $n = |V(G)|$ and $m = |E(G)|$. Moreover, the symbol T will be used to refer to rooted tree patterns and denote its size with $k = |V(T)|$. We will abuse terminology and use 'tree' for 'rooted tree' if it is clear from the context.

5 Mining Frequent Rooted Trees

In order to realise a pattern miner for rooted trees in single networks, the two most important ingredients are efficient generation of candidate trees and frequency counting of candidates in the network (using subgraph isomorphism). We proceed by first outlining our candidate generation method, and then reviewing the subgraph isomorphism method of [20] and showing how it can be employed to compute the above defined frequency measure. Finally we give complexity bounds for our complete miner.

5.1 Candidate Generation

In our miner, we use the same technique for generating rooted trees as in [25,26]. It generates rooted ordered candidate trees that are *left heavy*, i.e. children of a node are ordered and each left sub-tree is larger than the right sub-tree according to their canonical form. The left heavy property avoids generating trees that are isomorphically equivalent.

The method works by adding nodes only to the right most path, and ensuring that the condition of left heavy subtree is met with each new added node. The method thus produces a new tree for each added node, and can do so with delay $O(k)$ for size k trees. The left heavy subtree condition is met by maintaining a canonical form for the trees, which for unlabeled case is just the depth sequence of nodes in pre-order traversal. If the trees are labeled, then vertex and edge labels are inserted. Figure 1 gives some example trees with their corresponding depth sequence to illustrate the technique.

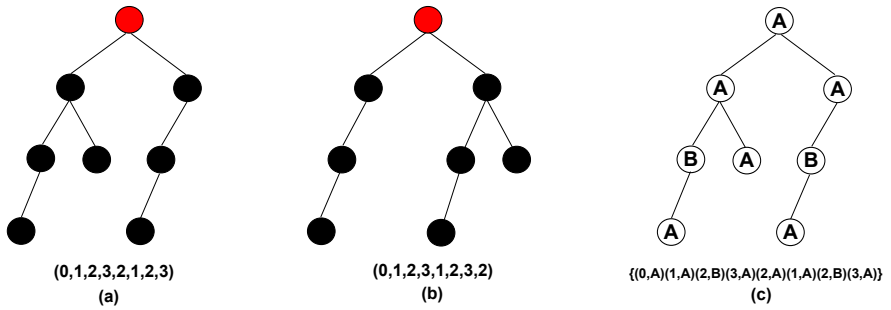


Fig. 1. The (rooted) unlabeled trees (a) and (b) are isomorphic to each other, but (a) is left-heavy compared to (b). We can treat as canonical form, the lexicographically heaviest string of pre-order traversal of depth sequence (given below each tree), and generate only trees that are like (a). Figure (c) shows an example of a left-heavy *labeled* tree with its corresponding canonical string.

5.2 Subgraph Isomorphism and Frequency Counting

Let us first briefly outline the subgraph isomorphism method of [20], which from here on we will call the Koutis&William's method. The method exploits the fact that for trees, subgraph homomorphisms can be computed in polynomial time. The method essentially consists of two core parts. In the first part, it constructs an arithmetic circuit computing a polynomial P representing all possible homomorphisms of a tree in a network. In particular, with every network vertex v a variable x_v is associated, and every homomorphism π from the pattern T to the network G corresponds to a term (monomial) $\prod_{v \in V(T)} x_{\pi(v)}$ in the polynomial, i.e. the product of the variables corresponding to the images of the vertices of the pattern. A multilinear term is a term where every variable occurs with degree at most 1. Isomorphisms are injective, and therefore the terms in P corresponding to isomorphisms will be exactly the multi-linear terms of P. In the second part, the method then evaluates the polynomial on an appropriate commutative group algebra, that ensures squares evaluate to 0. Hence, all terms which are not multi-linear (i.e. all homomorphisms which are not isomorphisms) vanish. The randomization of the values for which the variables x_v are substituted is such that multi-linear terms evaluate to non-zero with probability at least $1/4$ and the randomization of the coefficients of the polynomial P is such that the summation of non-zero monomials evaluates to non-zero with probability at least $7/8$.

In particular, [20] evaluates the polynomial P over $GF(2^l)\mathbb{Z}_2^k$. $\mathbb{Z}_2^k$ contains all bitvectors of length k. For $x, y \in \mathbb{Z}_2^k$, the multiplication is defined by componentwise addition of the elements of the bit vectors. The neutral element, the vector containing k zeros, is denoted W_0. Then, $GF(2^l)\mathbb{Z}_2^k$ is the ring of linear combinations of elements of $\mathbb{Z}_2^k$ with coefficients from $GF(2^l)$, the unique field with 2^l elements. $GF(2^l)$ has characteristic 2, i.e. $x + x = 0$ holds for any x.

The polynomial is evaluated by assigning to each variable x_v a value $W_0 + y_v$ where y_v is a random value from $\mathbb{Z}_2^k$ (i.e. a random k-bit vector). The result of [20] is based on the following observations:

- For a set $S \subseteq V(G)$ and variables $x_v = W_0 + y_v$ with $y_v \in \mathbb{Z}_2^k$, it holds that $\prod_{v \in S} x_v \neq 0$ iff the multiset $\{y_v | v \in S\}$ is a set of linearly independent vectors. Non-multilinear terms therefore evaluate to 0.
- A set of k randomly chosen bitvectors of length k is independent with probability at least $1/4$.
- For any set of element $b_i \in GF(2^l)\mathbb{Z}_2^k$ and randomly chosen coefficients $a_i \in GF(2^l)$, if any of the b_i is non-zero, then $\sum_i a_i b_i$ is nonzero with probability $1/2^l$.

Figure 2 gives an illustration of the above concept. It shows the mapping of a rooted tree to a network, and the corresponding polynomial for this mapping. The two multi-linear terms $x_1 x_2 x_3$ in the polynomial represent isomorphisms, while the rest represent homomorphisms.

In Algorithm 1, we outline the subgraph isomorphism method of [20]. The **occur** method in Algorithm 1 defines an arithmetic circuit of a polynomial for all homomorphic mappings starting from the mapping of root $r \in T$ to some $v' \in G$. The creation and evaluation of such circuits for all $v' \in G$, in method **countFreq**, gives us our above defined support measure of root images, for our root $r \in T$.

The $a_{r,j}$ and x_j, in the **occur** method, are chosen randomly from $\mathbb{Z}_2^k$ and $GF(2^l)$, and the arithmetic on the elements of array $C_{i,j}$ is performed based on their defined group algebra. In our implementation we use the representation-theoretic technique similar to [19] for doing the evaluations is memory linear in k (rather than linear in 2^k).

As per [20], the theoretical space complexity of **occur** method is $O(km)$ and its time complexity is $O(k^2 m 2^k l^2)$[1]. By extension the time complexity of **countFreq** method would be $O(k^2 mn 2^k l^2)$. However, we note that it is possible with only a single evaluation of the arithmetic circuit to obtain the values $\mathbf{occur(}\boldsymbol{T,G,r,j}\mathbf{)}$ for all j, and hence the time complexity is only $O(k^2 m 2^k l^2)$.

Remark 1. The **occur** method's success probability $p > 1/5$ can be increased to an arbitrary p', by repeating the method $\lceil log(1-p')/log(1-p) \rceil$ times (thereby decreasing the probability of failure) .

5.3 Complete Miner and Complexity Bounds

Algorithm 2 gives pseudo-code for our complete pattern miner. It brings together all the core components to make an apriori style miner.

We now proceed to prove its theoretical bounds. Note, the space complexity of our miner is bounded by our main datastructure $C_{i,j}$, and is $O(kn)$.

Theorem 1. *Given a network G with n nodes and m edges labeled by label set Σ_G, and a frequency threshold t we can mine all frequent tree patterns of size $\leq k$ with time $O(|\Sigma_G| \log^2(k) k^2 m 2^k)$.*

[1] $O(km)$ being the size of the circuit and $O(k 2^k l^2)$ to do arithmetic over $GF(2^l)[\mathbb{Z}_2^k]$.

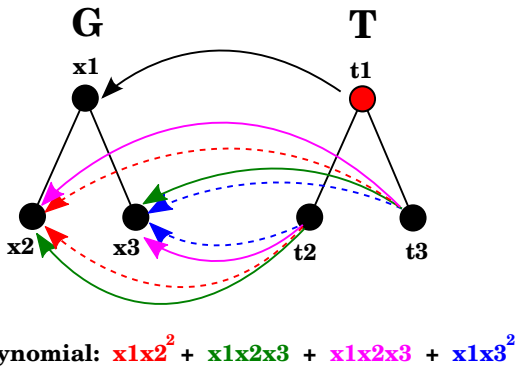


Fig. 2. All homomorphisms of T unto G, when mapping of $t1 \to x1$ is fixed. Solid lines represent mappings that are also isomorphic

Proof. As mentioned earlier, the **countFreq** method takes $O(\log^2(k)km2^k)$ time for each candidate tested. At each call made at level $i \geq 2$ the generateCandidates function in algorithm 2 produces at most $O(|\Sigma_G|)$ candidates per frequent pattern in S_{i-1}. For each such candidate, **countFreq** is called, taking time $O(\log^2(k)k^2m2^k)$. Therefore, for each new solution output, the algorithm will need $O(|\Sigma_G| \log^2(k)k^2m2^k)$ more time to finish. The theorem follows by noting that we can delay the printing of the solutions (frequent patterns) in such a way that between the printing of each pair of consecutive solutions the time is bounded by $O(|\Sigma_G| \log^2(k)k^2m2^k)$.

5.4 Further Optimizations

A number of optimizations are still possible with algorithm 2. Below we mention some of the more high level optimizations we implemented in our miner.

- **Sharing common subtrees.** Note that for each C_i for $i \geq 2$, the candidate trees may share a number of subtrees common among them. We do not need to test each candidate tree in isolation. Instead we can reuse the previously computed results of the common subparts.
  In our implementation we made a directed acyclic graph structure to represent all $T \in C_i$, i.e. a tree is represented by a node of the directed acyclic graph (the root) and all nodes below it. Several nodes can be parents of the same node and hence the corresponding trees can share subtrees. This method shares computation at the expense of additional memory.
- **Checking for homomorphisms.** As mentioned earlier, homomorphisms for trees in networks can be found in polynomial time. In fact in our case all we have to do is evaluate our circuit over the infinite field of integers, instead of the group algebra $GF(2^l)[\mathbb{Z}_2^k]$, thereby avoiding all the expensive arithmetic. Evaluation is linear in the size of the circuit and is only $O(km)$. We can store and use results of these inexpensive tests, and avoid the more expensive isomorphism tests for any part of network and common subtrees

Algorithm 1. Count frequency of tree T in a network G

```
let C_{i,j} be an array containing the result of mapping each v_i ∈ V_T to each v_j ∈ V_G

function occur(T, G, r, j):
if C_{r,j} is filled then
    return C_{r,j}
else if λ_T(v_r) ≠ λ_G(v_j) then
    let C_{r,j} := 0.
else
    let C_{r,j} := a_{r,j}.x_j {where x_j is a randomly chosen k-bit vector and a_{r,j} a random scalar}
    if |V(T)| > 1 then
        let S_{T'} := {subtrees after removing v_r from T}
        C_{r,j} := C_{r,j} · ∏_{T'∈S_{T'}} ( Σ_{j':(v_j,v_{j'})∈E_G} occur(T', G, r', j') ) .
    end if
end if
return C_{r,j}

function countFreq(T, G):
let: v_r ∈ V_T be the root of T
freq := 0
for j = 1 to |V_G| do
    if occur(T, G, r, j) then
        freq = freq + 1
    end if
end for
return freq
```

that are not homomorphic. In case of labeled networks especially, this can offer considerable speedup.

6 Experimental Evaluation

6.1 Experimental Setup

In our experimental evaluation, we are interested in the following experimental questions:

Q1 What size of patterns and networks can our algorithm handle within reasonable time?
Q2 How does our pattern matching strategy compare to state of the art strategies, in particular with VF2[8]?
Q3 Does our implementation scale as well as Koutis-William's theoretical algorithm?
Q4 What is the influence of pattern mining parameters and optimizations?

Algorithm 2. Find all patterns of size upto k

```
1: function findPatterns(G, k, t, Σ):
2: T := ∅ {set of all frequent trees}
3: S_{1...k} := ∅ {frequent trees of size 1...k}
4: C_{1...k} := ∅ {candidate trees of size 1...k}
5: for i = 1 to k do
6:    if i = 1 then
7:       C_1 := {single vertex graphs labeled with a label in Σ}
8:    else
9:       C_i := generateCandidates(S_{i-1}, Σ)
10:   end if
11:   S_i := {c_j : c_j ∈ C_i ∧ countFreq(c_j, G) ≥ t}
12:   T := T ∪ S_l
13: end for
14: return T
```

To perform our experiments, we implemented a system which we will call MINT (MIning Networks for Trees), containing a breadth-first pattern mining algorithm using the candidate generation method outlined in Section 5.1. We implemented the frequency counting based Koutis&William's algorithm as described in Section 5.2, and a baseline MINT-VF2 using frequency counting based on the VF2 algorithm [8]. We consider several versions of our new algorithm: First, MINT-STD implements a vanilla version of Koutis&William's algorithm. Second, MINT-HOMO implements Koutis&William's with homomorphism checking optimization, and third, MINT-BATCH which includes homomorphism checking as well as sharing common subtrees among the candidates. We call the last one MINT-BATCH, as we share subtrees only among *batchsize* number candidates in each pass; otherwise the memory requirements get intractably large due to the exponential number of frequent patterns.

In order to be able to compare the randomized algorithm to the deterministic VF2, the subgraph isomorphism tests were repeated a sufficient number of times to achieve a very high probability of success $(1 - 10^{-6})$. The result was that in all cases except one, the randomized algorithm found the same set of frequent patterns as the deterministic one (the only exception was MINT-STD which classified one out of $124,687$ frequent patterns of size 7 as infrequent for the 10^2 network in Table 2).

6.2 Data Sets

We present results on both synthetic as well as real-world data.

For synthetic data we generated power-law graphs with degree distribution $P(d) \propto d^{-4}$. Such graphs show significant clustering, as is often seen in real-world data. We generated networks of size $n = \{10^2, 10^3, 10^4, 10^5, 10^6, 10^7\}$, and then randomly assigned 1 of 4 labels to each of the vertices.

Table 1. Real datasets' summary

Dataset	# vertices	# edges	# vertex labels	# edge labels	Avg. degree
Facebook-uniform	984,830	185,508	17	1	0.38
Facebook-mhrw	957,359	1,792,188	16	1	3.74
Dblp-9202	129,073	277,081	1	11	4.29
Dblp-0305	109,044	233,961	1	3	4.29
Dblp-0507	135,116	290,363	1	3	4.28
IMDB	30,835,467	53,686,381	144	1	1.74

For real-world data, we used the DBLP citation network[2], the Facebook social network[3], and the IMDB movie database[4]. The DBLP data is a snapshot of their citation network from 1992-2007. It is the same data as was used in [1]. The Facebook data is the Facebook social network obtained through random sampling (one through uniform sampling, and the other through independent Metropolis-Hastings random walks [13]). For IMDB, we extracted the movie-actor network from the raw database. Our extracted network consists of movie, year, role and actor nodes. Movie and role nodes were labeled by movie and role type, whereas year nodes were labeled by the year the movie was released in. Actor nodes are left with a default label. Also, Table 1 gives basic statistics of our real-world networks.

6.3 Results

Complete mining of synthetic data. We ran the algorithms on synthetic datasets, and mined for as large patterns as we could in 10 hours. Table 2 gives the number of frequent patterns found in that time period for frequency threshold **0.1**, as a function of the network size and pattern size. It is noteworthy that as the network size grows, due to the asymptotic properties of the powerlaw graphs the number of frequent patterns of a given size converges. Figure 3 plots for each network size the total time used against the pattern size, for each of the considered algorithms. Note, that we could not run the MINT-BATCH for larger networks, due to its large memory requirements.

Sampled frequent patterns of synthetic data. The number of patterns grows exponentially. Nevertheless, large patterns may be of interest. A strategy which gained popularity recently [14] is to not mine all frequent patterns but only generate a sample of them. Here, we adopt a simple sampling strategy of randomly selecting only 100 frequent patterns at each level (denoted pattern size in our breadth-first mining) of the mining process, to make extensions for the next level. This experiment allows us to study more closely the delay (time used per pattern found) of our miner.

[2] http://www-kdd.isti.cnr.it/GERM/
[3] http://odysseas.calit2.uci.edu/doku.php/public:online_social_networks
[4] http://www.imdb.com/interfaces

Table 2. Number of frequent patterns for synthetic data

network size / pattern size	10^2	10^3	10^4	10^5	10^6	10^7
1	3	2	2	2	2	2
2	10	4	4	4	4	4
3	48	22	22	22	22	22
4	295	144	142	142	142	142
5	2077	1076	1066	1066	1066	
6	15,698	8605	8534	8534		
7	124,687	72084				
8	1,024,557					

Table 3. Number of frequent patterns for real data

network / pattern size	FB-uniform	FB-mhrw	Dblp0305	Dblp0507	Dblp9202	IMDB
1	2	1	1	1	1	6
2	1	2	3	3	8	10
3	2	5	12	13	10	38
4	3	11	51	57	10	149
5	4	30	189	277	6	692
6	5	88	648	1099	1	
7	10				0	
8	15					

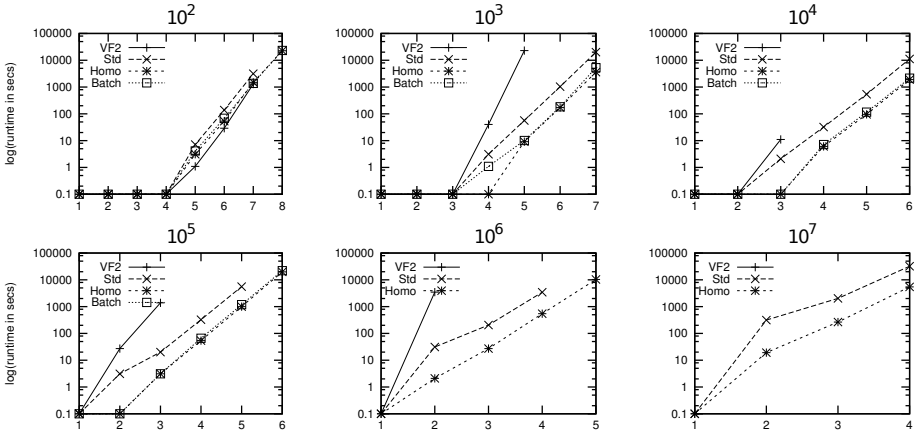


Fig. 3. log-runtime as a function of pattern size

Figure 4 plots for each network the delay (time used per pattern) as a function of the size of the patterns, and also as a function of the size of the network for patterns of size 4, for each of the considered algorithms.

Real-world datasets. Here we followed essentially the same procedure as for synthetic data, the only differences being that a smaller frequency threshold of 0.05

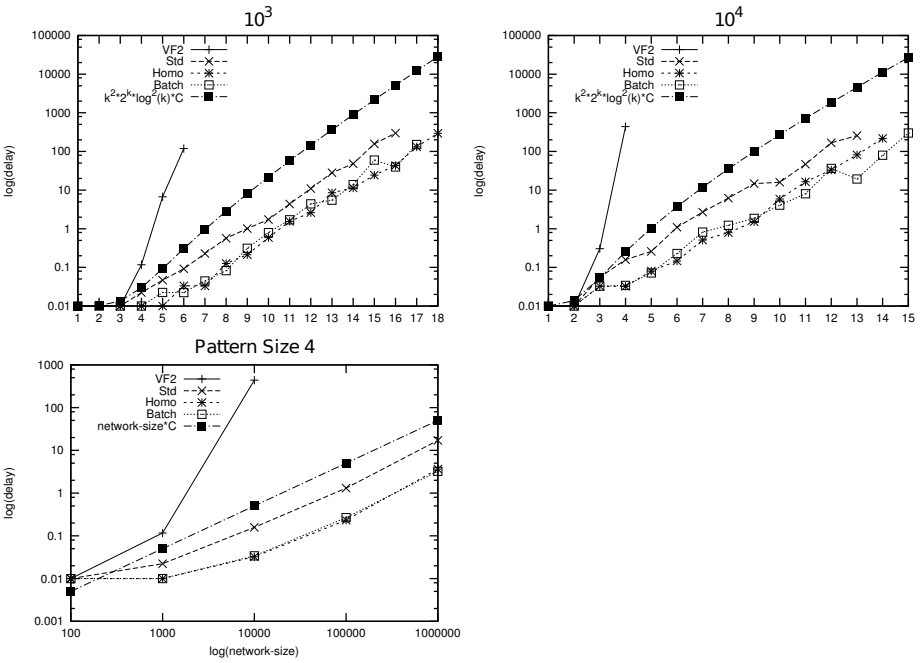


Fig. 4. log-delaytime (time per pattern) as a function of pattern size, as well as a function of network size

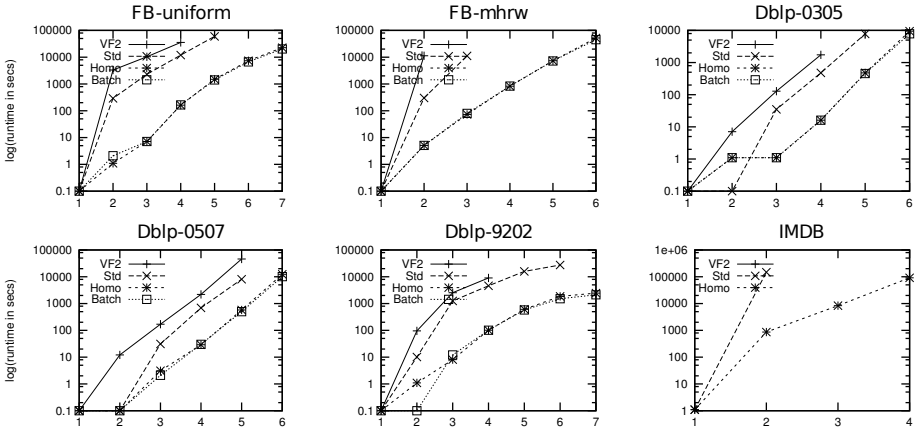


Fig. 5. log-runtime as a function of pattern size

for FB-uniform and IMDB was used to allow for larger patterns to be mined, and that a higher cut-off point for runtime was used (we allowed 16 hours for DBLP, 24 hours for Facebook, and 48 hours of runtime for IMDB data). Table 3 lists the number of frequent patterns found for each network. Figure 5 plots for each network the total time used against the pattern size, for each of the considered algorithms.

6.4 Discussion

Based on the results reported above, we can answer the experimental questions as follows:

Q1 Using the new pattern matching method, it is computationally feasible to match patterns up to size 15 (see figure 4). The main bottleneck when mining all patterns is the number of frequent patterns found. As this number increases exponentially, in many settings we don't get further than size 5 patterns. One can observe however, that for real-world pattern mining tasks one often has prior domain knowledge allowing for pruning the search space towards the type of patterns one is interested in.

Q2 It is clear from all experiments that the new pattern method is orders of magnitude better than the VF2 algorithm, especially for larger patterns.

Q3 From figure 4 one can see that the pattern matching algorithm scales at least as well as the theoretical upper bound. In particular, in contrast to VF2, our new method scales linearly in the network size and scales indeed as $O(k^2 \log^2(k) 2^k)$ in the pattern size.

Q4 The homomorphism check prunes away a significant amount of subgraph isomorphism tests for patterns which are clearly infrequent. This especially holds for the real-world dataset.

7 Conclusion and Future Work

We present a novel algorithm for mining trees in single networks. It scales well with respect to network size, and is only mildly exponential in pattern size, which makes it tractable for moderately sized patterns. We show the effectiveness of the method in practice, on real as well as synthetic data.

As for future work, we expect that several heuristic optimizations are possible which can improve performance on real-world datasets. Furthermore, we would also like to extend our method to graph classes other than trees.

Acknowledgements. This research is supported by ERC Starting Grant 240186 "MiGraNT: Mining Graphs and Networks, a Theory-based approach". We thank Anton Dries and Constantin Commendant for the valuable suggestions.

References

1. Berlingerio, M., Bonchi, F., Bringmann, B., Gionis, A.: Mining Graph Evolution Rules. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part I. LNCS, vol. 5781, pp. 115–130. Springer, Heidelberg (2009)
2. Bogdanov, P., Mongiovì, M., Singh, A.K.: Mining heavy subgraphs in time-evolving networks. In: Proceedings of the 2011 IEEE 11th International Conference on Data Mining, ICDM 2011, pp. 81–90. IEEE Computer Society, Washington, DC (2011)

3. Borgelt, C., Berthold, M.R.: Mining molecular fragments: Finding relevant substructures of molecules. In: Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2002, pp. 51–58. IEEE Computer Society, Washington, DC (2002)
4. Borgelt, C., Meinl, T., Berthold, M.: Moss: a program for molecular substructure mining. In: Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, OSDM 2005, pp. 6–15. ACM, New York (2005)
5. Bringmann, B., Nijssen, S.: What is frequent in a single graph? In: Frasconi, P., Kersting, K., Wrobel, S. (eds.) Proceedings of MLG-2007: 5th International Workshop on Mining and Learning with Graphs, pp. 1–4 (2007)
6. Calders, T., Ramon, J., Van Dyck, D.: All normalized anti-monotonic overlap graph measures are bounded. Data Mining and Knowl. Disc. 23(3), 503–548 (2011)
7. Chi, Y., Xia, Y., Yang, Y., Muntz, R.: Mining closed and maximal frequent subtrees from databases of labeled rooted trees. IEEE Trans. on Knowl. and Data Eng. 17, 190–202 (2005)
8. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: An improved algorithm for matching large graphs. In: 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen, pp. 149–159 (2001)
9. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 1367–1372 (2004)
10. Diestel, R.: Graph Theory, 4th edn., electronic edn. Springer (2010)
11. Dries, A., Nijssen, S.: Mining Patterns in Networks using Homomorphism. In: Proceedings of the Twelfth SIAM International Conference on Data Mining, pp. 260–271. Omnipress (April 2012), https://lirias.kuleuven.be/handle/123456789/350328
12. Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C.: Using ghost edges for classification in sparsely labeled networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, pp. 256–264. ACM, New York (2008)
13. Gjoka, M., Kurant, M., Butts, C., Markopoulou, A.: Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In: Proc. of IEEE INFOCOM 2010 (2010)
14. Hasan, M.A., Zaki, M.J.: Output space sampling for graph patterns. Proceedings of the VLDB Endowment 2(1), 730–741 (2009)
15. Henderson, K., Gallagher, B., Li, L., Akoglu, L., Eliassi-Rad, T., Tong, H., Faloutsos, C.: It's who you know: graph mining using recursive structural features. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2011, pp. 663–671. ACM, New York (2011)
16. Huan, J., Wang, W., Prins, J.: Efficient mining of frequent subgraphs in the presence of isomorphism. In: Proceedings of the 2003 Third IEEE International Conference on Data Mining, ICDM 2003, pp. 549–556. IEEE Computer Society, Washington, DC (2003)
17. Huan, J., Wang, W., Prins, J., Yang, J.: Spin: mining maximal frequent subgraphs from graph databases. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2004, pp. 581–586. ACM, New York (2004)
18. Inokuchi, A., Washio, T., Motoda, H.: An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)

19. Koutis, I.: Faster Algebraic Algorithms for Path and Packing Problems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 575–586. Springer, Heidelberg (2008)
20. Koutis, I., Williams, R.: Limits and Applications of Group Algebras for Parameterized Problems. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 653–664. Springer, Heidelberg (2009)
21. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM 2001, pp. 313–320. IEEE Computer Society, Washington, DC (2001)
22. McKay, B.D.: Practical graph isomorphism. Congr. Numerantium 10, 45–87 (1981)
23. Nienhuys-Cheng, S.-H., de Wolf, R.: Foundations of Inductive Logic Programming. LNCS (LNAI), vol. 1228. Springer, Heidelberg (1997)
24. Nijssen, S., Kok, J.: There is no optimal, theta-subsumption based refinement operator, personal communication
25. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2004, pp. 647–652. ACM, New York (2004)
26. Nijssen, S., Kok, J.N.: The gaston tool for frequent subgraph mining. Electronic Notes in Theoretical Computer Science 127(1), 77–87 (2005); Proceedings of the International Workshop on Graph-Based Tools (GraBaTs 2004)
27. Thomas, L.T., Valluri, S.R., Karlapalem, K.: Margin: Maximal frequent subgraph mining. ACM Trans. Knowl. Discov. Data 4, 10:1–10:42 (2010)
28. Ullmann, J.: An algorithm for subgraph isomorphism. JACM 23(1), 31–42 (1976)
29. Wörlein, M., Meinl, T., Fischer, I., Philippsen, M.: A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 392–403. Springer, Heidelberg (2005)
30. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2002, pp. 721–724. IEEE Computer Society, Washington, DC (2002)
31. Yan, X., Han, J.: Closegraph: mining closed frequent graph patterns. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, pp. 286–295. ACM, New York (2003)

Reachability Analysis and Modeling of Dynamic Event Networks

Kathy Macropol and Ambuj Singh

Department of Computer Science
University of California
Santa Barbara, CA 93106 USA
{kpm,ambuj}@cs.ucsb.edu

Abstract. A wealth of graph data, from email and telephone graphs to Twitter networks, falls into the category of dynamic "event" networks. Edges in these networks represent brief events, and their analysis leads to multiple interesting and important topics, such as the prediction of road traffic or modeling of communication flow. In this paper, we analyze a novel new dynamic event graph property, the "Dynamic Reachability Set" (DRS), which characterizes reachability within graphs across time. We discover that DRS histograms of multiple real world dynamic event networks follow novel distribution patterns. From these patterns, we introduce a new generative dynamic graph model, DRS-Gen. DRS-Gen captures the dynamic graph properties of connectivity and reachability, as well as generates time values for its edges. To the best of our knowledge, DRS-Gen is the first such model which produces exact time values on edges, allowing us to understand simultaneity across multiple information flows.

Keywords: Graph Generator, Dynamic Networks, Reachability.

1 Introduction

Vast amounts of graph datasets are generated each day by applications such as social networks, communication networks (like email graphs or Twitter), bioinformatics, and the Internet. The analysis and mining of these networks has been an active and important area of research, leading to both newly discovered fundamental network properties, as well as interesting and useful new knowledge and applications, from graph clustering for gene function discovery to network modeling for link or structure prediction [24,25,26].

Previous work on the analysis of graphs and their properties have analyzed degree distribution, number of triangles, relationships between the eigenvalues of the graph, etc [14,33,31]. These discovered properties have led to novel generative graph models, capable of producing new graph structures which capture and mimic such properties. Generative graph models have many interesting and important uses, including generation of synthetic datasets for analysis, graph anonymization, graph compression, prediction of graph and link evolution. While useful for many purposes, these graph models still mimic only the static network

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 442–457, 2012.

graph structure. The addition of dynamic components to these static networks has been an active topic of research in the last few years, with much research concentrated on dynamic "state" networks (where links have a tendency to endure) such as in social or collaboration networks.

In contrast to dynamic state networks, links represent brief events in dynamic "event" networks. Examples of such graphs include communication networks like email or telephone call graphs. Dynamic event networks form a large and important category for graph datasets. However, their structure and dynamics do not fit well with many of the current dynamic network models. In dynamic event networks, timestamps associated with each link may convey the dynamics, allowing for both evenly spaced flows as well as arbitrarily long pauses or bursts of activity. This timed dynamic behavior is an integral part of a time-evolving network, and can be difficult to capture in models.

In this paper, we focus on dynamic event networks, looking especially at a new property, the "Dynamic Reachability Set" (DRS), which characterizes reachability within graphs across time. Reachability in general, along with concepts such as graph density or planarity, is a fundamental network property. In an evolving graph, reachability can convey latency of information flow between pairs of nodes in dynamically changing networks (e.g. mobile ad hoc or sensor networks); or latency along logistic/supply chain networks under dynamics; or even gossiping latency in dynamic social networks. Specifically, the DRS of a starting node consists of the set of all nodes reachable from the starting node, across a fixed time interval Δ. In this paper, we analyze the DRS properties, at a series of time intervals, for nodes within multiple real world dynamic event networks. From this analysis, we discover that DRS sizes follow a DGX distribution [6] for low values of Δ, but that this relation breaks as Δ increases. Additionally, we find that the rate in which the relation changes is specific to each network, but generally follow a log-normal-like curve.

The dynamic behaviors discovered from our analysis open the door for the learning and creation of new, novel generative modeling techniques which can now link time together with changes in network structure. Using this discovery, we focus on the generation of network dynamics, and propose a new generative modeling algorithm, DRS-Gen, able to produce dynamic graph structures that mimic the DRS properties of real world dynamic event networks across time. To the best of our knowledge, DRS-Gen is the first generative graph model able to assign and fit timestamps to edges, such that the rates of flow and reachability across a dynamic network are preserved. We introduce methods to learn the model parameters, and from there implement DRS-Gen, fitting this model to multiple real world dynamic event networks. From our results, we find that our generated graphs fit well and capture the flow and reachability distributions of real world graphs, across time, making it both a novel and potentially useful tool for generative graph modeling and analysis.

Overall, our main contributions come in three parts. First, we introduce and analyze a novel, relevant, and interesting new dynamic event graph property: the Dynamic Reachability Set. Second, we propose a new generative dynamic event

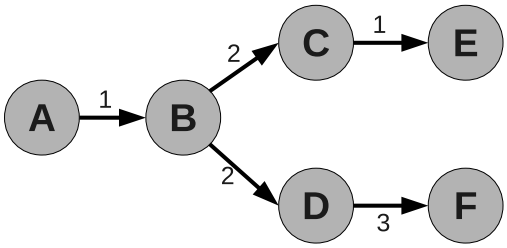


Fig. 1. An example dynamic event graph. The numbers on edges represent timestamps for the links. A time threshold of $\Delta = [1, 2)$ would give node A a DRS of {A,B}; for $\Delta = [1, 3)$, DRS = {A,B,C,D}; for $\Delta = [1, 4)$, DRS = {A,B,C,D,F}.

graph model, DRS-Gen, that allows for the generation of time-aware edges and flows. And third, we demonstrate not only how DRS-Gen may produce graphs, but also how it can be fit, naturally and easily, to real-world communication graphs, such that it may capture the reachability and dynamics of these graphs.

The rest of this paper is organized as follows. Section 2 introduces the concept of Dynamic Reachability Sets, as well as contains the analysis and results obtained from studying these sets on real world graphs. Section 3 introduces the DRS-Gen model, outlining the algorithm and theory behind it, then presenting and analyzing the results of our model when fit to multiple real world dynamic networks. Section 4 consists of a short survey of related work and previously introduced techniques on graph analysis and generation. Finally, in Section 5, we summarize our work, overviewing our contributions and conclusions.

2 Dynamic Reachability Sets

For this work, we focus our attention on the property of reachability within dynamic event networks. Specifically, let $G = \{V, E, T\}$ be a directed, dynamic graph where V are the set of vertices and $E = [e_1, e_2, \cdots, e_m]$ are the list of edges, where edge $e_i = (v_j, v_k)$ represents a link between nodes v_j and v_k. Additionally, the edges in E are ordered by the time function T, where $T(e_i)$ gives the timestamp for edge e_i. We can then define the DRS starting from node v_s and timestamp t_{start}, recursively over time interval Δ, as shown in Algorithm 1.

Algorithm 1. Calculate the DRS

Require: $\mathbf{DRS} := \{v_s\}, t_{start}, t_{end} := t_{start} + \Delta$
1: **procedure** CalcDRS($\mathbf{DRS}, t_{start}, t_{end}$)
2: **for all** v_i **where** $(e_k = (v_s, v_i), v_s \in \mathbf{DRS}, e_k \in E, t_{start} \leq T(e_k) < t_{end})$
3: $\mathbf{DRS} := \mathbf{DRS} \cup \{v_i\}$
4: **CalcDRS**($\mathbf{DRS}, T(e_k) + 1, t_{end}$)
5: **end procedure**

Figure 1 illustrates an example. In this graph, directed edges are associated with timestamps, and node A sends a message at timestamp 1 to node B. Node

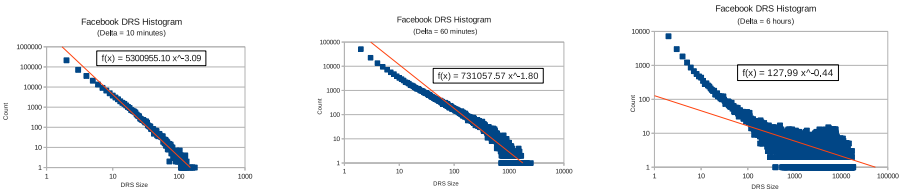


Fig. 2. Histograms of DRS size for the Facebook wall posts dataset, at increasing values for Δ. The decreasing slope of the approximately fit power law curve shows the movement of "mass" to the right as Δ increases.

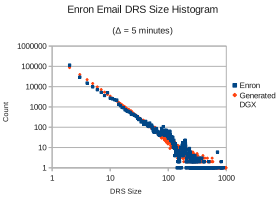


Fig. 3. Histogram of DRS size for the Enron Email dataset at $\Delta = 5$ minutes, along with the a generated DGX distribution ($\sigma = -7.67$, $\mu = 3.03$)

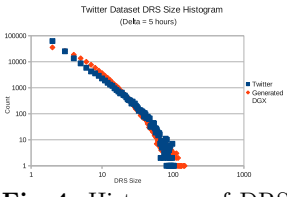


Fig. 4. Histogram of DRS size for the Twitter dataset at $\Delta = 5$ hours, along with the a generated DGX distribution ($\sigma = 1.10$, $\mu = 0.990$)

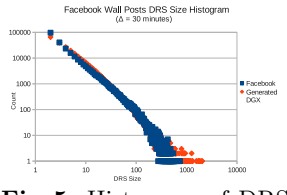


Fig. 5. Histogram of DRS size for the Facebook Wall post dataset at $\Delta = 30$ minutes, along with the a generated DGX distribution ($\sigma = 0.013$, $\mu = 1.88$)

B sends two messages, one to node C and one to node D at timestamp 2, etc. To find the DRS of node A, given the time interval Δ, we collect the set of all nodes that may be reached by recursively traveling edges within the time frame, starting from node A, without going backwards in time. This means that an interval starting at 1, with a Δ of 2, will include nodes A, B, C, D,and F, but not node E because to reach node C from node A takes until timestamp 2, and the edge from C to E occurred previously at timestamp 1.

The reachability set of a node, n_s, represents the nodes it is possible for n_s to reach across a specific time interval. Furthermore, the sequence of nodes and links followed to obtain the DRS can be thought of as a small "flow" within the graph. Overall, the set of DRS values for all nodes in a graph provides a window into the graph's dynamic connectivity and flow between all of its nodes. As the DRS time interval grows, we would expect the average DRS sizes to increase as well, since the number of links contained within the interval, and therefore the chances of reaching additional nodes, grows as well.

We collected and analyzed the actual DRS sets for multiple real world networks, and found that this intuition does indeed hold. Figure 2 shows the log-log plot of DRS size versus count, for a network consisting of Facebook wall posts and replies [34] containing 47K nodes (users) and 877K directed edges (wall posts from one user to the other). For each time interval, a series of non-overlapping time windows (for different values of t_1, the initial timestamp mentioned in Equation (1)) was used to discover DRS counts for each node in the graph. From the plots, we can see that as Δ increases from 10 minutes to 6 hours, more

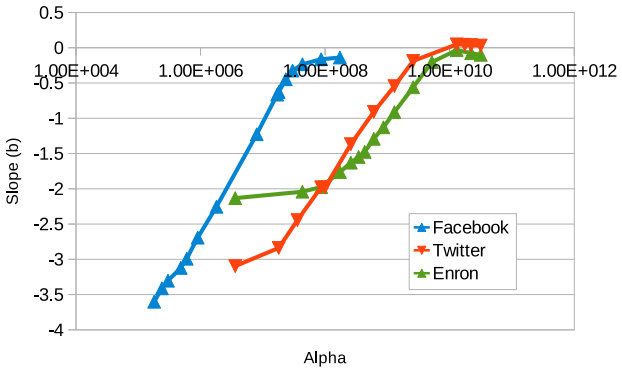


Fig. 6. Comparison of slope (for fit log-log curve) against Δ (logarithmic). It can be seen that various graphs each have their own rate of change. However, each follows a reversed log-normal form.

nodes gather on the right of the plot. Similar curves were found for other dynamic networks, as well (Figures 9 and 11 display additional histogram results for the Enron Email dataset [15] consisting of 87K unique email users and 360K directed email edges, as well as a crawled Twitter dataset [26] containing 8K Twitter users and 663K directed "reply to" and "retweet" edges). Again, this shift across time is largely reflected in the plots, and is echoed in the decreasing slope of the power-law curve loosely fit to the data. As the time interval increases, more nodes are able to reach larger amounts of the graph. Additionally, we can see that the shape of the curve undergoes an extreme change across time, as well. Additionally, it is apparent that the rate of change in the slope and curves varies, depending upon the graph. Figure 6 shows a plot of the slopes for the various networks, as they change across time, confirming that each network has its own properties related to reachability and flow, producing different network behavior.

Overall, the DRS histograms have points that cluster tightly along a curve for smaller values of Δ, but eventually "pile" to the right as the maximum, or near maximum, number of nodes they may add is reached.

It has previously been discovered that Discrete Gaussian Exponential (DGX) distributions match well to many real world datasets [6], and fitting a DGX distribution to the curves obtained at small Δ values for the DRS, it can be seen from Figures 3, 4, and 5, that this distribution matches the DRS histogram at low Δ threshold values, as well.

An interesting insight can be discovered by plotting the negative of the log-log slopes from the power law distributions for each DRS histogram, such as those in Figure 2. The resulting curve for the Enron Email network can be seen in Figure 7. The negative of this log-log slope fits well to a log-normal curve, a property echoed in each of the other dynamic event networks analyzed, as well.

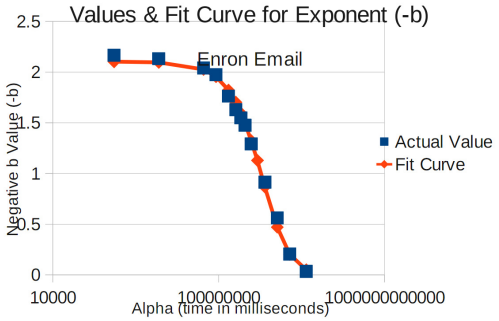


Fig. 7. Fitting a log-normal curve for the value of (-b), the negative exponent in the power law curve

3 The DRS-Gen Model

As a model, DRS-Gen focuses upon the generation and modeling of network dynamics and flow. A wealth of work in previous literature has focused upon the creation of static graph generators [4,7,10,17,18,20,28], and so, rather than repeat their work, DRS-Gen instead assumes that a base (static) graph structure is available. This structure could be the original network itself, without the dynamics and multi-edges, or it could be generated using any one of the many existing static network graph generators. For the experiments in this work, we use the original networks as a base, and generate dynamic behavior upon it.

The basis behind DRS-Gen is the shift in "weight" that occurs as the time interval Δ is increased. This shift can be seen for example in the DRS histogram plots of Figure 2, where a much larger number of points have collected to the right for the last graph in Figure 2, as compared to the preceding graph. This shift represents the fact that, as the time interval increases, more nodes are able to reach a larger section of the graph, giving them an increased DRS size. This property allows us to relate graph structure together with time. For any given time interval Δ, the associated DRS histogram essentially counts multiple small flows, separated by either time or graph structure. As the time interval increases, these smaller flows may join together to become a single larger flow, contributing toward the shift in mass, as two smaller flows are replaced by a new larger flow in the histogram.

To model this shift, DRS-Gen takes 5 parameters as input: a min and max time resolution Δ_{min} and Δ_{max}, a starting number of flows c, and two parameters, μ and σ which model the change in slope of a log-log power-law distribution fitted to the normalized DRS histogram, across time. It then proceeds in four basic steps.

1. First, we generate a series of c number of integers, representing the DRS sizes for a set of initial "base flows."
2. Next, we transform this series of integers into a series of small subgraph structures representing the flows.

3. We then search through the subgraphs, finding and combining subgraphs which overlap by choosing a time, δ_t, that represents the time differences between the occurrences of the flows.
4. Finally, we output the final generated graph. Initial flows are given random timestamps, and the δ_t values are used to calculate the timestamps for overlapping flows.

We describe these four steps in more detail in the following subsections.

3.1 Generating Flow Sizes

Given our minimum time resolution Δ_{min}, we want to generate a series of c integers. These integers represent the DRS size of our c initial "base flows". The series of DRS sizes should fit the appropriate normalized DRS histogram distribution, which we model using a power law curve, following the form:

$$pr[\mathbf{DRS}] = a(\mathbf{DRS})^{-b} \tag{1}$$

Where **DRS** stands for a particular DRS size. In this case, the exponent b represents the slope of the line arising within the log-log plot, as can be seen by taking the logarithm of both sides of Equation (1).

$$\ln(pr[\mathbf{DRS}]) = \ln(a) - b\ln(\mathbf{DRS}) \tag{2}$$

As the time interval (Δ) is increased, the amount of mass in the curve shifts to the right, resulting in a variation for b across time. Figures 6 and 8 showed that b varies across Δ and fits well to a log-normal distribution, which is represented by:

$$b = \frac{1}{\Delta\sigma\sqrt{2\pi}} e^{-\frac{(\ln\Delta-\mu)^2}{2\sigma^2}} \tag{3}$$

where μ and σ are the location and scale parameters of the fit log-normal curve.

Equation (1), together with Equation (3), can be used to relate the probability of seeing a DRS / flow size, against a particular time interval.

Since we wish to obtain a series of DRS sizes, **fSizes**, at base time resolution Δ_{min}, we substitute Δ_{min} for Δ into these equations, to obtain the probability distribution we wish to achieve. We find and sample from the inverse of the CDF of this probability distribution to obtain a similar distribution. The CDF becomes:

$$\begin{aligned} F_x &= \int_2^x ax^{-b} \\ &= \frac{a}{1-b}\left(x^{1-b} - 2^{1-b}\right) \end{aligned} \tag{4}$$

We invert it and obtain:

$$F_y = \left(\frac{(1-b)x}{a} + 2^{1-b}\right)^{\frac{1}{1-b}} \tag{5}$$

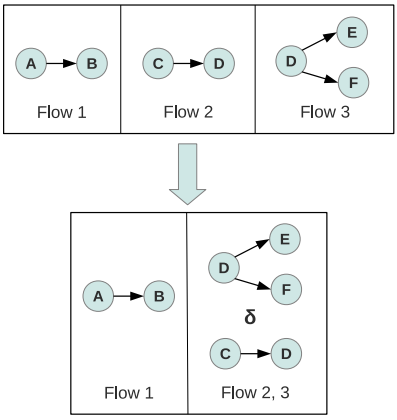


Fig. 8. Example of flows in F combining. Here, Flow 3 (of size 3 nodes) overlaps with Flow 2 (of size 2 nodes), and will therefore be combined, using a time difference δ between them, into a flow with 4 unique nodes.

Additionally, by assuming an approximate bound on the maximum DRS size, taken from the x intercept of the log-log plot shown in Equation (2) (at $\sqrt[b]{a}$), we can obtain a relationship for a using the pdf from Equation (1).

$$\begin{aligned} 1 &= \int_2^{\sqrt[b]{a}} ax^{-b} \\ &= \frac{\sqrt[b]{a} - 2^{1-b}a}{1-b} \end{aligned} \tag{6}$$

We can calculate the value for a from Equation (6) using Newton's method. After sampling from Equation (5), we then obtain our series of DRS flow sizes, **fSizes.**

3.2 Obtaining Subgraph Flow Structures

In order to obtain the subgraph flow structures, the series of integers found in the previous section must be transformed into a series of small subgraph structures, $F = [F_1, F_2, \cdots, F_c]$ of corresponding size. Drawn from the given base network, each subgraph will represent a single flow. There are multiple methods which can be used to create these small, static initial subgraph structures. We choose to use a simple variation (simply enforcing connected subgraphs) on the "Winners Don't Take All" method [29], which grows subgraphs by repeatedly choosing connected nodes either through random chance or through preferential attachment. The method used (random or preferential attachment) is randomly chosen at each step, as well.

3.3 Combining Overlapping Flows

With the series of base graph flows discovered, the next step is to combine overlapping flows. To do this, we assume a tentative ordering in time on F, and systematically search through the flows in F. For each flow, F_i, in F, we find all preceding flows, starting from F_1 and working our way up, which also overlap in graph structure. For every discovered overlapping pair of flows, we choose a time difference δ, using a probability function, that represents the amount of time that passes between their occurrence. These two flows are then combined (the time difference δ between them is noted) and the process continues. Figure 8 contains an example of this process, where Flows 2 and 3 overlap. A time difference δ is chosen between them, and they are combined. Pseudocode for this process is contained in Algorithm 2.

Given that we have found two overlapping simple flows of sizes S_1 and S_2, they will combine at some point in time, producing a single flow (of size S_3, where $max(S_1, S_2) \leq S_3 \leq S_1 + S_2 - 1$). The time difference between them is represented by δ. The two possibilities (whether the flows are combined or not) can be represented as two separate distributions: R, a distribution representing the combined state and having a probability of 1 for flow size S_3 and 0 for every other flow size, and U, a distribution representing the uncombined state and having a probability U_1 for flow size S_1, U_2 for S_2, and U_3 for S_3 (with U_1, U_2, and U_3 being discrete values of either 0, 0.5, or 1). As an example, in Figure 8, Flow 2 has a size of $S_1 = 2$ and Flow 3 has a size of $S_2 = 3$. When they combine, they produce a flow with size $S_3 = 4$. The probability distribution for the combined state, R, has probability 1 for size 4, and a probability of 0 for every other size. The distribution for the uncombined state, U, has probabilities 0.5 for size 2, 0.5 for size 3, 0 for size 4, as well as 0 for every other size.

The distance between these two possible distributions, and the modeled "true" distribution, T, of Equation (1) may be calculated. The likelihood of the flows combining may be found by comparing the distance between R and T with the distance between U and T. For distance comparison, we choose to use the Kullback-Leibler divergence (KL divergence).

$$\begin{aligned} D_{KL}(P||Q) &= \sum_{i \in S} P(i) \ln \frac{P(i)}{Q(i)} \\ &= \sum_{i \in S} P(i)(\ln P(i) - \ln Q(i)) \end{aligned} \tag{7}$$

From Equation (7), it can be seen that the KL divergence calculates a weighted distance between corresponding points on the log-log curve. This means that a distribution with a closer distance is more likely.

The probability values for the modeled distribution T can be found by using Equation (1), and are normalized.

$$T_i = \frac{P(S_i)}{\sum_{j \in S} P(S_j)} \tag{8}$$

Algorithm 2. DRS-Gen

```
1: procedure GENERATE(F)
2:     Initialize fSizes[ ]
3:     for i ← 1, c do
4:         r ← random number from Eq. 6
5:         fSizes[i] ← r
6:     Initilize F[ ]
7:     for i ← 1, c do
8:         Initialize K[ ]
9:         for j ← 1, fSizes[i] do
10:            K[j] ← new node using
11:              "Winners Don't Take All"
12:        F[i] ← K
13:    Initilize Deltas[ ][ ]
14:    for i ← 1, c do
15:        for j ← i − 1, 1 do
16:            if F[i] ∩ F[j] > 0 then
17:                if Deltas[i][j] isn't set then
18:                    δ ← random number from Eq. 13
19:                    Deltas[i][j] = δ
20:                    for k ← 1, c do
21:                        if Deltas[j][k] exists then
22:                            Deltas[i][k] = δ + Deltas[j][k]
23: end procedure
```

Substituting using Equation (11), we obtain

$$T_i = \frac{aS_i^{-b}}{\sum_{j \in S} aS_j^{-b}} = \frac{S_i^{-b}}{\sum_{j \in S} S_j^{-b}} \tag{9}$$

$$\left(where\ b = \frac{1}{\Delta\sigma\sqrt{2\pi}} e^{-\frac{(\ln \Delta - \mu)^2}{2\sigma^2}} \right)$$

When b is found using the parameters μ and σ, we obtain an equation relating the probability of certain sized flow occurring, to the time interval Δ.

To find the likelihood of combining, we calculate the relative closeness:

$$Pr[combining] = 1 - \frac{D_{KL}(R||T)}{D_{KL}(R||T) + D_{KL}(U||T)} \tag{10}$$

Substituting using Equation (7), we obtain Pr[combining] as:

$$1 - \frac{\sum_{i \in S} R_i(\ln R_i - \ln T_i)}{\sum_{i \in S} R_i(\ln R_i - \ln T_i) + \sum_{i \in S} U_i(\ln U_i - \ln T_i)} \tag{11}$$

Given that $R_1 = 1$, all other $R_i = 0$, and at least one of S_1, S_2, or $S_3 = 0$, we may simplify and combine with Equation (11), obtaining Pr[combining] as:

$$1 - \frac{b \ln S_1 + \ln(S_1^{-b} + S_2^{-b} + S_3^{-b})}{b \ln(S_1^{1+U_1} S_2^{U_2} S_3^{U_3}) + 2 \ln(S_1^{-b} + S_2^{-b} + S_3^{-b})} \tag{12}$$

where b is the log-normal curve shown in Equation (3), and all other values are constant. This probability varies across time, as Δ varies between Δ_{min} and Δ_{max}. A δ value, weighted by this distribution, can be picked using a series of approximations. The overall function and its integral can be approximated using Taylor series. Additionally, the area under the curve is calculated using the given value for Δ_{max}. Next, a random fraction of this AUC is chosen, and finally the appropriate δ for this AUC can be solved for numerically using Newton's method.

From this process, values for δ between overlapping flows are generated, with δ values fitting the dynamic distribution behavior discovered in Section 2.

3.4 Producing the Generated Graph

With the flow and timing information generated, these structures can be output to produce the final graph. Starting with the first flow F_1, a timestamp of 0 is assigned and output. Next, all other flows combined with F_1 are output, using their assigned time differences to produce their timestamps. If any flows remain, they are assigned a random timestamp, output, and their connected flows output as before. This process repeats until all flows have been output.

3.5 Parameter Fitting

From the steps in the overall algorithm, it is a simple extension to fit this generator to a known graph. First, a series of DRS histograms, at varying Δ, are calculated. Next, the c values for each histogram are normalized, producing a probability distribution. A power curve is fit to each distribution and the values for the power, b, extracted. From this series of b values, the appropriate values of μ and σ can be estimated by fitting a log-normal curve, and the generator may now be fit to the graph, using these parameter values.

Overall, this process allows a dynamic event graph to be generated, with dynamic reachability behavior fit to parameters learned from real world data.

3.6 Implementation and Analysis

Using this method, we implemented DRS-Gen and used the the captured DRS histograms for the Enron Email, Facebook wall post, and Twitter networks mentioned earlier and shown in Figures 2 and 3, to train (using the parameter fitting methods described in Section 3.5) a generative model capable of producing flows which imitate the properties of the original, real world graphs.

Figures 9, 10, and 11 show the resulting DRS histograms obtained through generation by the model, as compared to the original distribution. As can be seen,

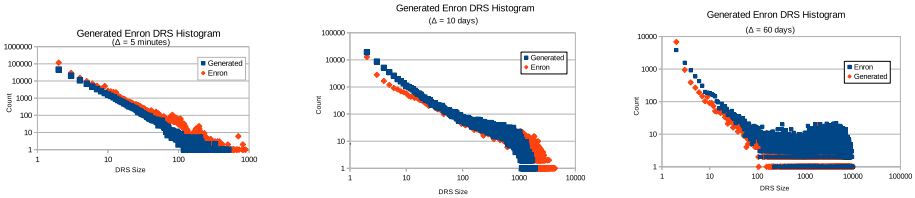


Fig. 9. Comparison of the DRS size histograms for both a graph generated using DRS-Gen, as well as the original Enron Email dataset, at increasing values for Δ. Both the tight fit of points, as well as the similarity in shape and dynamics emphasize the strength and quality of DRS-Gen's dynamic modeling results.

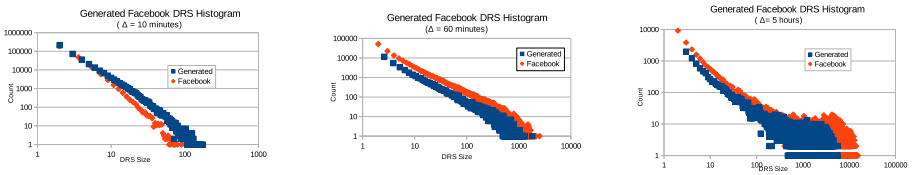


Fig. 10. Comparison of the DRS size histograms for both a graph generated using DRS-Gen, as well as the original Facebook dataset, at increasing values for Δ. Again, the strong similarity between both the original histogram and the generated, across time, helps to confirm the effectiveness of DRS-Gen's model.

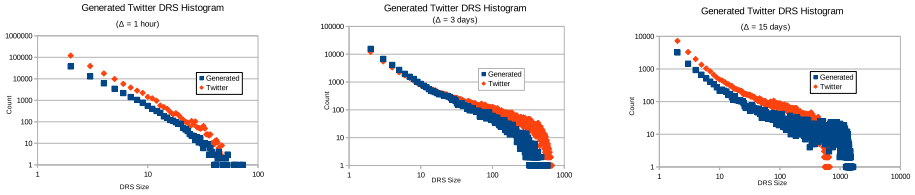


Fig. 11. Comparison of the DRS size histograms for both a graph generated using DRS-Gen, as well as the original Twitter dataset, at increasing values for Δ

both the distribution shapes, slopes, as well as the rates of change across time match extremely well to the original dynamic network distributions. Though each of the three original networks evolve at different rates, the graphs generated from our model manage to capture this evolution and fit the generated flows together in time such that the flow distribution and reachability match closely to the original curve.

A series of Quantile-Quantile plots are shown in Figure 12, comparing the original and generated distributions for the Enron Email, Facebook wall post, and Twitter graphs (at Δ =5 minutes, 10 minutes, and 1 hour respectively). The close fit to a straight y = x line further emphasizes the closeness of the generated vs. original distributions, and helps to confirm that our generative model is truly capturing the distribution and reachability of the original network.

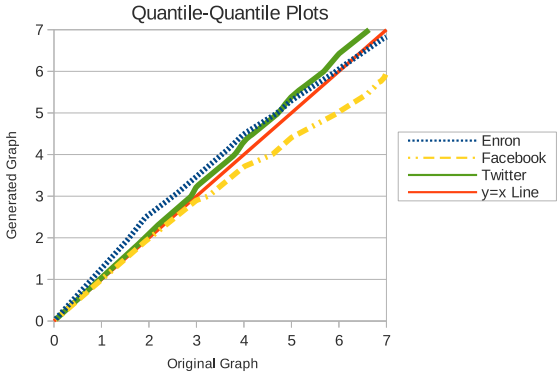


Fig. 12. Quantile-Quantile plots of the Enron Email, Facebook wall post, and Twitter graphs. The close fit to the y=x line emphasizes the closeness of the generated vs. original distributions.

These results help to confirm the effectiveness of our model, emphasizing its strength and ability to generating dynamic event network data, while capturing the dynamic reachability and flow properties of the original graph.

4 Related Work

Decades of research on graph theory has concentrated on studying fundamental properties of graphs and been successfully applied to the analysis and modeling of graph data and real world networks; however, researchers have mainly looked at static graph properties, leading to generative models that mimic static network structure [3,11,20]. In contrast, temporal graphs have been an active topic of research in only the last few years and the research has largely concentrated on dynamic "state" networks and ignored dynamic "event" networks [2,26,32].

A major focus of current research is on generative models that allow for the prediction of the slow evolution (long-term dynamics) of graph structure [12,16]. Previously introduced models for dynamic graphs include the Markovian Dynamic Graph models, which are random models where the graph structure at every time step t is dependent only on the structure at time $t-1$, and created according to random transition probabilities. In Edge-Markovian Dynamic Graphs [9], each edge at time step t is dependent only on its presence (or not) at $t-1$. There are fixed global birth and death rate functions, giving the probability of a new edge arising and an old edge dying. A variation on this model, where nodes are initially assigned a fixed position, and node distances affect birth and death rate values, was introduced in [13].

Despite their elegant formulation and ease of analysis, the Markovian Dynamic Graph models fail to capture many real-world network properties. For example, two general dynamic graph properties that have been observed are densification power laws, relating the number of nodes and edges of a graph over time to a power law distribution, as well as shrinking diameters across time [21]. To capture these discovered properties, new generative graph models were introduced.

On example is the Forest Fire model, where new links are formed by randomly choosing "ambassador" nodes and recursively following their links, linking to discovered nodes with a certain probability [21]. Other dynamic network properties recently discovered include the bursty-weight law, where edge weight additions were found to be bursty over time [27,19], and the relation between age of a node and its likelihood to attract new edges [20]. From these observations, new generative models such as the Butterfly graph model and Triangle-closing models were introduced [19,27]. Another recent graph generative model which accounts for numerous static as well as dynamic graph properties is RTG [1], based on the concept of random typing. In RTG, a set of keys have a probability distribution representing their likelihood to be typed. Every word randomly typed is a node label, and the stream of nodes typed are divided into source and destination pairs to create edges.

Dynamic processes on complex networks such as information diffusion and epidemiological processes have also been studied [5]. Epidemic models, such as the Susceptible-Infected-Susceptible (SIS) model [3], have been applied to the modeling of link cascades within blogs in [22,23]. Interestingly, even though the process is time-varying, the network in these models are usually considered static or changing very slowly. In contrast, recent work by Prakash et. al [30] analyzes virus propagation graphs by formulating them as an approximate nonlinear dynamical system. In [8], the authors utilize the spectral radius of the adjacency matrix for predicting the virulence of epidemics on static graphs.

Additionally, most current dynamic network generative models use the concept of abstract "timesteps" for their dynamics, which do not fit well with real world event graphs. Typically, for many models, each timestep label corresponds to a single event rather than a precise measure of time. This sequence of labels conveys the network dynamics. However, many real world networks instead have actual time values associated with each link, allowing for both evenly spaced flows as well as arbitrarily long pauses or bursts of activity. This timed behavior is an integral part of a dynamic network, and its oversight leaves a large area of important graph data and knowledge largely unexplored. The lack of time values upon edges also renders it difficult to calculate and compare DRS values from graphs generated by these algorithms to the original graphs, as Δ intervals cannot be easily mapped onto timesteps.

5 Summary

In this paper, we have introduced and analyzed a novel new property of dynamic event networks, their Dynamic Reachability Sets (DRS), across time. The DRS characterizes reachability within a graph across time, and connects to many important graph relationships such as network flow and latency, in addition to reachability. From this analysis, we have discovered several important new properties of dynamic networks, including a novel distribution pattern for the DRS histograms, related to a DGX distribution at small time intervals.

Additionally, we have made use of this newly discovered pattern by introducing a new generative graph model, DRS-Gen, based upon the DRS distribution

dynamics. DRS-Gen is capable of generating event network dynamics upon graphs, and particularly able to fit naturally to real world event networks, learning parameters that can capture and model dynamic flow and reachability across time. Dynamic graph models such as DRS-Gen can have many possible practical applications, including prediction of future graph evolution or behavior (such as in link or email thread prediction), and graph compression.

Implementing DRS-Gen and testing it on multiple networks, we find that the generated graphs closely matched the distributions and dynamics of the original networks they modeled, helping to emphasize DRS-Gen's use and effectiveness as a new dynamic event network graph generator, and a novel and potentially useful new tool for generative graph modeling and analysis.

Acknowledgements. Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative 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 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.

Work was also partially supported by the National Science Foundation under grant IIS-0917149.

References

1. Akoglu, L., Faloutsos, C.: RTG: A Recursive Realistic Graph Generator Using Random Typing. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part I. LNCS, vol. 5781, pp. 13–28. Springer, Heidelberg (2009)
2. Akoglu, L., Mcglohon, M., Faloutsos, C.: Rtm: Laws and a recursive generator for weighted time-evolving graphs. In: ICDM 2008 (2008)
3. Bailey, N.: The mathematical theory of infectious disease and its applications. Hafner Press (1975)
4. Barabási, A.-L., Albert, R.: Emergence of scaling in random networks. Science 286(5439), 509–512 (1999)
5. Barrat, A., Barthlemy, M., Vespignani, A.: Dynamical Processes on Complex Networks, New York, NY, USA (2008)
6. Bi, Z., Faloutsos, C., Korn, F.: The "dgx" distribution for mining massive, skewed data. In: KDD, pp. 17–26 (2001)
7. Chakrabarti, D., Faloutsos, C.: Graph mining: Laws, generators, and algorithms. ACM Comput. Surv. 38 (June 2006)
8. Chakrabarti, D., Wang, Y., Wang, C., Leskovec, J., Faloutsos, C.: Epidemic thresholds in real networks. ACM Trans. Inf. Syst. Secur. 10, 1:1–1:26 (2008)
9. Clementi, A.E., Macci, C., Monti, A., Pasquale, F., Silvestri, R.: Flooding time in edge-markovian dynamic graphs. In: PODC, pp. 213–222 (2008)
10. Erdös, P., Rényi, A.: On the evolution of random graphs. In: Publication of the Mathematical Institute of the Hungarian Academy of Sciences, pp. 17–61 (1960)
11. Fabrikant, A., Koutsoupias, E., Papadimitriou, C.: Heuristically Optimized Trade-Offs: A New Paradigm for Power Laws in the Internet. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, p. 110. Springer, Heidelberg (2002)

12. Goldenberg, A., Zheng, A.X., Fienberg, S.E., Airoldi, E.M.: A survey of statistical network models. Found. Trends Mach. Learn. 2, 129–233 (2010)
13. Grindrod, P., Higham, D.J.: Evolving graphs: dynamical models, inverse problems and propagation. Proc. of TRSA: Math, Phys. Engr. Sci. 466, 753–770 (2010)
14. Kleinberg, J.M., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.S.: The web as a graph: Measurements, models, and methods (1999)
15. Klimt, B., Yang, Y.: Introducing the enron corpus. In: CEAS (2004)
16. Kuhn, F., Oshman, R.: Dynamic networks: models and algorithms. SIGACT News 42, 82–96 (2011)
17. Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., Upfal, E.: Stochastic models for the web graph. In: Proc. Found. of CS, pp. 57–66 (2000)
18. Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tompkins, A., Upfal, E.: The web as a graph. In: PODS, pp. 1–10. ACM, New York (2000)
19. Leskovec, J., Backstrom, L., Kumar, R., Tomkins, A.: Microscopic evolution of social networks. In: KDD (2008)
20. Leskovec, J., Chakrabarti, D., Kleinberg, J.M., Faloutsos, C.: Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 133–145. Springer, Heidelberg (2005)
21. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: KDD (2005)
22. Leskovec, J., Mcglohon, M., Faloutsos, C., Glance, N., Hurst, M.: Cascading behavior in large blog graphs: Patterns and a model. Technical report (2006)
23. Leskovec, J., McGlohon, M., Faloutsos, C., Hurst, M.: Cascading behavior in large blog graphs patterns and a model. In: SDM (2007)
24. Macropol, K., Can, T., Singh, A.: Rrw: repeated random walks on genome-scale protein networks for local cluster discovery. BMC Bioinformatics 10, 283 (2009)
25. Macropol, K., Singh, A.: Scalable discovery of best clusters on large graphs. PVLDB 3(1), 693–702 (2010)
26. Macropol, K., Singh, A.K.: Content-based modeling and prediction of information dissemination. In: ASONAM (2011)
27. McGlohon, M., Akoglu, L., Faloutsos, C.: Weighted graphs and disconnected components: patterns and a generator. In: KDD, pp. 524–532 (2008)
28. Nickel, C.L.M.: Random Dot Product Graphs: A Model For Social Networks. PhD thesis, Johns Hopkins University, Maryland, USA (2006)
29. Pennock, D., Flake, G., Lawrence, S., Glover, E., Giles, C.L.: Winners don't take all: Characterizing the competition for links on the web. In: PNAS (2002)
30. Prakash, B.A., Tong, H., Valler, N., Faloutsos, M., Faloutsos, C.: Virus Propagation on Time-Varying Networks: Theory and Immunization Algorithms. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part III. LNCS, vol. 6323, pp. 99–114. Springer, Heidelberg (2010)
31. Siganos, G., Faloutsos, M., Faloutsos, P., Faloutsos, C.: Power laws and the as-level internet topology. IEEE/ACM Trans. Netw. 11, 514–524 (2003)
32. Snijders, T.A., van de Bunt, G.G., Steglich, C.E.: Introduction to stochastic actor-based models for network dynamics. Social Networks 32, 44–60 (2010)
33. Tsourakakis, C.E.: Fast counting of triangles in large real networks without counting: Algorithms and laws. In: ICDM (2008)
34. Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in facebook. In: WOSN 2009 (2009)

CC-MR – Finding Connected Components in Huge Graphs with MapReduce

Thomas Seidl, Brigitte Boden, and Sergej Fries

Data Management and Data Exploration Group
RWTH Aachen University, Germany
{seidl,boden,fries}@cs.rwth-aachen.de

Abstract. The detection of connected components in graphs is a well-known problem arising in a large number of applications including data mining, analysis of social networks, image analysis and a lot of other related problems. In spite of the existing very efficient serial algorithms, this problem remains a subject of research due to increasing data amounts produced by modern information systems which cannot be handled by single workstations. Only highly parallelized approaches on multi-core-servers or computer clusters are able to deal with these large-scale data sets. In this work we present a solution for this problem for distributed memory architectures, and provide an implementation for the well-known MapReduce framework developed by Google. Our algorithm CC-MR significantly outperforms the existing approaches for the MapReduce framework in terms of the number of necessary iterations, communication costs and execution runtime, as we show in our experimental evaluation on synthetic and real-world data. Furthermore, we present a technique for accelerating our implementation for datasets with very heterogeneous component sizes as they often appear in real data sets.

1 Introduction

Web and social graphs, chemical compounds, protein and co-author networks, XML databases - graph structures are a very natural way for representing complex data and therefore appear almost everywhere in data processing. Knowledge extraction from these data often relies (at least as a preprocessing step) on the problem of finding connected components within these graphs. The horizon of applications is very broad and ranges from analysis of coherent cliques in social networks, density based clustering, image segmentation, where in some way connected parts of the image have to be retrieved, data base queries and many more. Thus, it is not surprising that this problem has a long research history, and different efficient algorithms were developed for its solution. Nevertheless, modern information systems produce more and more increasing data sets whose processing is not manageable on single workstations any more. Social networks like Facebook process networks with more then 750 million users[1] where each

[1] http://www.facebook.com/press/info.php?statistics, state Sep. 2011.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 458–473, 2012.

node is connected to 130 other nodes on average. The analysis of such enormous data volumes requires highly scalable parallelized algorithms. In this paper, we present a highly scalable algorithm for MapReduce [4], which is a programming model for the development of parallel algorithms developed by Google Inc. in 2004. Since then it experienced a fast spread out and nowadays its open-source implementation Hadoop[2] is used in companies like Yahoo! Inc. or Facebook Inc..

In MapReduce, the data is given as a list of records that are represented as (key, value) pairs. Basically, a MapReduce program consists of two phases: The first phase is the "Map" phase, in which the records are arbitrarily distributed to different computing nodes (called "mappers") and each record is processed separately, independent of the other data items. In the second phase, called "Reduce" phase, records having the same key are grouped together and processed in the same computing node ("reducer"). Thus, the reducers combine information of different records having the same key and aggregate the intermediate results of the mappers. By using this framework, programmers may concentrate on the data flow which is implemented by map jobs and reduce jobs. They do not have to take care of low-level parallelization and synchronization tasks as in classic parallel programming.

On top of this new programming model, Hadoop and other implementations of the MapReduce framework show a lot of non-functional advantages: First, they are scalable to clusters of many computing nodes, which are easily expanded by new nodes. Moreover, they show a high fault-tolerance: If one of the computing nodes fails during the execution of the program, the work of the other nodes is not affected or discarded, instead just the records that were currently processed on the failing node have to be processed again by another node.

In this paper, we propose an algorithm for finding connected components which is based on the MapReduce programming model and is implemented using Hadoop. Thus, our approach can make use of the aforementioned advantages of Hadoop such as high scalability and fault-tolerance.

The main contributions of our paper are:

- We present the parallelized algorithm CC-MR for the efficient detection of connected components in a graph using the MapReduce framework.
- We evaluate the performance of our algorithm compared to state-of-the-art approaches using synthetic and real-world datasets.
- We develop a technique to improve the load balancing of CC-MR for graphs with heterogeneous component sizes.

2 Fundamentals

In this section we give a short formal problem definition for the finding of connected components in Section 2.1. In Section 2.2 we introduce the MapReduce framework, which is used for the implementation of our algorithms.

[2] http://hadoop.apache.org/

2.1 Connected Components

Let $G = (V, E)$ be an undirected graph without self loops, with V being a set of vertices and $E = \{(v, u), (u, v)\}, u, v \in V$ a set of edges. Intuitively, a connected component in G is a maximal subgraph $S = (V^S, E^S)$ in which for any two vertices $v, u \in V^S$ there exists an undirected path in G with v as start and u as end vertex. The term "maximal subgraph" means that for any additional vertex $w \in V \setminus V^S$ there is no path from any $v \in V^S$ to w.

In this work we present a solution for finding all connected components inside the graph G. The algorithm can as well be applied to directed graphs, in this case the result is the set of all weak connected components in the graph.

2.2 MapReduce

MapReduce is a programming model for processing web-scale datasets presented by Deam and Ghemawat at Google [4]. The main idea of this model is to divide data processing into two steps: map and reduce. The map phase is responsible for processing given (key,value) pairs stored on the distributed file system and generating intermediate (key,value) pairs. In the reduce phase, intermediate pairs with the same key are collected, processed at once, and the results are stored back to the distributed file system.

In the MapReduce model, communication between different computing nodes only takes place during a single communication phase, when the intermediate pairs from the map nodes are transferred to the reduce nodes. Apart from this, no further communication takes place. Neither do the individual mappers nor the individual reducers communicate with each other. This loose coupling of the computational nodes enables the framework to perform the calculations in a highly distributed and fault-tolerant way. Since all computational nodes process the data independently from each other, the only limitation for the number of parallel reducer-jobs is the number of unique intermediate key values. Additionally, since the single jobs do not depend on the results of other jobs, the failure of hardware can be easily managed by restarting the same job on another computational node. This high fault-tolerance and the loose coupling of computational nodes suits perfectly for usage of this model on commodity hardware like personal PCs connected to a cluster over a network. However, this limited communication also poses a challenge for the development of algorithms, which have to be designed such that the data in different mappers/reducers can be processed completely independently. Especially, the results computed by different reducers can not be combined in a MapReduce job. Thus, many problems cannot be solved using a single MapReduce job, but have to be solved by a chain of MapReduce jobs, such that the result records of one iteration can be re-distributed to the reducers and thus combined in the next iteration.

3 Related Work

The detection of connected components in a graph is a fundamental and well-known problem. In the past, different approaches for finding connected

components were introduced. The diversity of the proposed techniques ranges from simple linear time techniques using breadth-first search / depth-first search to efficient logarithmic algorithms. Though, due to the fast growing data sizes (just think of social network graphs of Facebook or Google+), even these efficient algorithms cannot deal with such big graphs. Thus, approaches to parallelize the detection of this components have already been developed for several decades: Hirschberg et al. [6] present an algorithm which uses n^2 processors (where $n = |V|$ denotes the number of vertices in the graph) and having a time complexity of $O(\log^2 n)$. Chin et al. [2] present a modified version of this algorithms which achieves the same time bound with only $n \left\lceil \frac{n}{\log^2 n} \right\rceil$ processors. A parallel algorithm with a time bound of $O(\log n)$ is presented by Shiloach and Vishkin [11]. Greiner [5] presents an overview of several parallel algorithms for connected components.

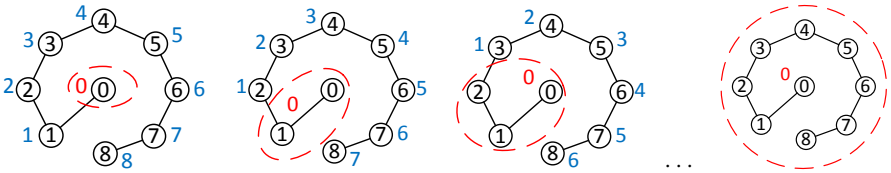
All of the aforementioned approaches assume that all computing processors have access to a shared memory and, thus, can access the same data. In contrast, the MapReduce model relies on distributing the data as well as the computation between the computing nodes and, thus, reduce the required communication between the computing nodes. There also exist some approaches that are based on a "distributed memory" model, i.e they consider the cost of communication between the computing nodes, e.g., the approach proposed in [1] which is an extension of the algorithm of [8]. In their distributed memory model, every computing node is able to access the memory of other computing nodes, which, however, leads to certain communication costs. In contrast, the MapReduce model only allows for special communication flows. E.g., communication between different reducers in a MapReduce job is not possible. Thus, for computing connected components using MapReduce, special types of algorithms are necessary.

Recently, a few approaches for the detection of connected components using the MapReduce model were proposed. Wu et al. [12] present an algorithm for detecting connected components based on Label Propagation. PEGASUS [7] is a graph mining system based on MapReduce and also contains an algorithm for the detection of connected components. In this system, graph mining operations are represented as repeated matrix-vector multiplications. In [9] the problem is solved by finding a minimum spanning tree of the graph. For that, edges which certainly do not belong to any MST are iteratively removed until the subgraphs are small enough to be processed by a single machine. Two further algorithms were proposed in [10]. These aims at minimizing the number of iterations and communication per step. The authors provide probable bounds which are logarithmic in the largest component size but claim that in practice the number of iterations for one of the algorithms is at most $2 \log d$ (d=diameter of the graph).

In [3] another connected components algorithm based on MapReduce is presented. As this algorithm is the most similar one to our approach, it will be introduced in the following. In this algorithm, nodes are assigned to so-called zones, where each zone is identified by the vertex with the smallest ID contained in this zone. Initially, each node defines an own zone. The zones are then merged iteratively until finally each zone corresponds to a connected component of the

graph: In each iteration, each edge is tested whether it connects nodes from different zones. Subsequently the algorithm finds for each zone z the zone z_{min} with the smallest ID that is connected to z and adds all vertices of z to z_{min}.

A drawback of this algorithm is that for each iteration, three MapReduce jobs have to be executed and in each iteration all edges of the original graph have to be processed.



(a) Initial graph (b) First Iter. (c) Second Iter. (d) Last Iter.

Fig. 1. Example for the algorithm from [3]

In Fig. 1 we show the processing of this algorithm for a simple example graph consisting of just one component. The numbers inside the vertices are the IDs of the vertices, the numbers beside the vertices denote the number of the zone a vertex is currently assigned to. The vertices that are already assigned to the "final" zone 0 are encircled. Initially, each vertex is assigned to its own zone. In the first iteration, the algorithm determines the edges that connect vertices from different zones, i.e. the zones 0 and 1. Then, i.e. for zone 1 the algorithm detects that the smallest zone connected to it is zone 0, i.e. the vertex 1 is now assigned to zone 0. Similarly, the vertex 2 is assigned to zone 1 etc.. In the second iteration, the same processing is done, i.e. vertex 2 is added to zone 1, vertex 3 (former zone 2) is added to zone 1 etc. Overall, the algorithm needs 8 iterations to detect the component. This example shows another drawback of the algorithm: Although in the first iteration e.g. the connection between zone 1 and zone 0 and the connection between zone 1 and zone 2 are detected, this information is not used in the second iteration. Using this information, we could e.g. directly add the vertex 3 from zone 2 to zone 0, as we know they are connected via zone 1 and thus have to belong to the same connected component. By neglecting these information, the algorithm needs a large number of iterations.

The basic idea of our new algorithm is to use this kind of information from previous iterations by adding additional edges ("shortcuts") in the graph such that fewer iterations are needed to find the final components. In our experimental section we compare our approach to the approaches from [3] and [7].

4 Algorithm

In this section we present our CC-MR-algorithm for detecting components in large-scale graphs using the MapRecuce framework. In subsection 4.1 we describe

our solution. For better understanding, we show in section 4.2 the processing of CC-MR on the example from section 3. Section 4.3 provides a formal proof of the correctness of CC-MR.

The basic idea of our algorithm is to iteratively alter growing local parts of the graph until each connected component is presented by a star-like subgraph, where all nodes are connected to the node having the smallest ID. For that, in each iteration, we add and delete edges such that vertices with larger IDs are assigned to the reachable vertex with smallest ID. Applying an intelligent strategy to use the information from previous iterations, our algorithm needs significantly less iterations than existing approaches.

4.1 CC-MR Algorithm

Basically there are two states for a (sub)component S: either it is already maximal or there are still further subcomponents which S can be merged with. The main question of every algorithm for finding connected components is, therefore, how to efficiently recognize those two states and how to react on them. I.e., if a component is already maximal, no further step should be performed, and in the second case the merging or some other equivalent action should be done with as little effort as possible. When dealing with parallel algorithms the question of balanced distribution of the calculations arises. Considering a distributed memory programming model like MapReduce additionally complicates the problem since an efficient information flow between independent computational nodes has to be established.

We propose a solution to handle the aforementioned states locally for every graph vertex in such a way that after at most linearly many iterations (experiments often show a logarithmic behavior for big graphs) in terms of the diameter of the largest component the solution is found. Pushing down the problem to the single vertices of the graph enables a very scalable processing in the MapReduce framework. Additionally, by using techniques for prevention of duplicated data, which often appears in distributed memory models, CC-MR-algorithm significantly outperforms the state-of-the art approaches as e.g. [3].

Let $G = (V, E)$ be an undirected graph where V is a set of vertices with IDs from $\mathbb{Z}$ and $E = \{(v_{source}, v_{dest}) \in V^2\}$ is a set of edges. The algorithm's basic idea is simple: independently check for each vertex v and its adjacent vertices $adj(v)$ whether v has the smallest ID or not. If yes (*locallyMaxState*), assign all $u \in adj(v)$ to v and stop the processing of v, since the component of v is already locally maximal. Otherwise (*mergeState*), there is a vertex $u \in adj(v)$ with $u < v$; then connect v and $adj(v)$ to u. This corresponds to assigning (merging) the component of v to the component of u. By iteratively performing these steps each component is finally transformed to a star-like subgraph where the vertex having the smallest ID is the center. The overall algorithm stops as far as the *mergeState* situation does not occur any more.

In the following, we present an efficient implementation based on a simple concept of *forward* and *backward edges*. We call an edge $v \to u$ a forward edge, if $v < u$, and a backward edge, if $v > u$, where the comparison of vertices means

the comparison of their IDs. Both types of edges are represented by a tuple (v, u) which we represent by $(key, value)$ pairs in the MapReduce framework. The semantic of the forward edge (v, u) is that vertex u belongs to the component of the vertex v. The backward edge can be regarded as a "bridge" between the component of vertex u and the component of a vertex w which has a connection to v as shown in Fig. 2. The backward edge between vertices v and u enables the reducer of v to connect u and w in a single iteration of the algorithm.

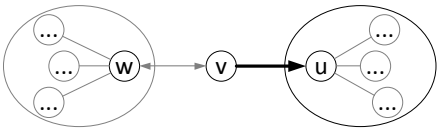


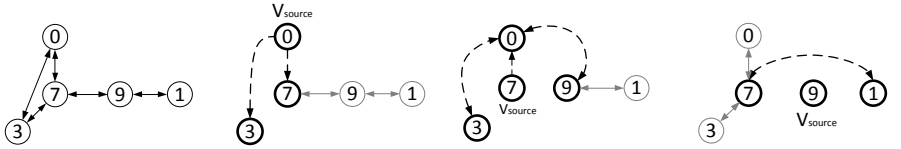
Fig. 2. The backward edge (v, u) connects components of w and u in the reducer of v

These concepts will become clearer from the explanation of the algorithm.

Listing 1.1. Reducer implementation.

```
1 newIterationNeeded = false // global variable
2 void reduce(int v_source, Iterator<int> values)
3     isLocMaxState = false
4     v_first = values.next();          // take first element
5     if ( v_source < V_first )
6         isLocMaxState = true
7         emit(v_source, v_first)
8     v_dest_old = v_first
9     while ( values.hasNext() )
10        v_dest = values.next()
11        if ( v_dest == v_dest_old ) continue // remove duplicates
12        if ( isLocMaxState )       // locMaxCase
13            emit( v_source, v_dest ) // only fwd. edge
14        else     // cases stdMergeCase, optimizedMergeCase
15            emit( v_first, v_dest ) // fwd. edge and
16            emit( v_dest, v_first ) // backwd. edge
17            newIterationNeeded = true
18        v_dest_old = v_dest
19    // stdMergeCase
20    if ( v_source < v_dest && !isLocMaxState )
21        emit( v_source, v_first ) // backwd. edge
```

As described earlier, a MapReduce job consists of a map and a reduce phase. In our case, the mapper is a so-called identity mapper which simply passes all read data to the reducer without performing any changes. The pseudo-code for the reduce phase is given in listing 1.1. The emitted edges (v_{source}, v_{dest}) are automatically grouped by their v_{source} values and then sorted in ascendent order



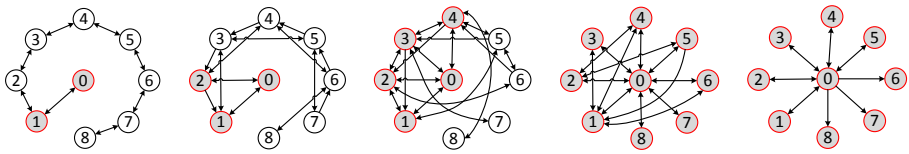
(a) Original graph (b) *locMaxCase* (c) *stdMergeCase* (d) *optimizedMergeCase*

Fig. 3. Three cases of the algorithm. v_{source} strings marks the node under consideration in the considered reducer. Bold highlighted nodes are adjacent nodes of v_{source}.

of their v_{dest}-values. Technically we use the secondary sort method of Hadoop to establish the desired sorting. The main part of the algorithm is located in the reducer (listing 1.1), where both aforementioned cases are handled. Tuples having the same key arrive as a data stream and are processed one after another in the 'while' loop. After the elimination of duplicate entries in the line 11, three cases are distinguished:

- *locMaxCase*: lines 5–7 and 12–13
- *optimizedMergeCase*: lines 15–18
- *stdMergeCase*: lines 15–18 and 20 – 21

locMaxCase corresponds to the *locallyMaxState*, i.e., it deals with the situation when a local maximal component with root v_{source} is already found and therefore all adjacent nodes $v_{dest} \in adj(v_{source})$ have to be assigned to v_{source}. This assignment is performed by emitting forward edges $v_{source} \rightarrow v_{dest}$ in the lines 7 and 13. Fig. 3(b) depicts the processing of the case *locMaxCase* by showing the changes of the original graph structure from Fig. 3(a). Nodes marked by v_{source} are the nodes which are considered in single reducer with all its adjacent nodes, which for their part are highlighted by bold circles. The dimmed circles show the remaining vertices of the graph which are not regarded during the computation of the node v_{source}. Dashed arrows represent the newly created edges inside the reducer. In this example the reducer of the node 0 emits therefore two edges $0 \rightarrow 3$ and $0 \rightarrow 7$. Cases *stdMergeCase* and *optimizedMergeCase* on their part deal with the merge state (*mergeState*), where *optimizedMergeCase* is a special case of *stdMergeCase* which reduces duplicate edges, as will be shown later. Both cases arise, if the condition $v_{source} > v_{first}$ holds, which means that at least one of the adjacent nodes $v_{dest} \in adj(v_{source})$ has a smaller ID than v_{source}. Due to the fact that the vertices are sorted in order of their IDs, v_{first} has the smallest value. Since the main aim of the algorithm is to assign all vertices with larger IDs to the vertex with smallest ID, this implies that all vertices in $adj(v_{source})$ except for v_{first} itself are assigned to v_{first}, i.e., for each of this vertices a forward edge $v_{first} \rightarrow v_{dest}$ (line 15) is emitted. Please note that this is not the case for the edge $v_{first} \rightarrow v_{source}$, since this edge will be emitted in the reducer of the vertex v_{first}. Therefore, in the example of *stdMergeCase* in Fig. 3(c) the edges $0 \rightarrow 3$ and $0 \rightarrow 9$ are emitted.



(a) Initial graph (b) Iteration 1 (c) Iteration 2 (d) Iteration 3 (e) Iteration 4

Fig. 4. Example for CC-MR

In addition to the forward edges, the algorithm emits backward edges $v_{dest} \rightarrow v_{first}$ (line 16), i.e., edges $3 \rightarrow 0$ and $9 \rightarrow 0$ in the example. These edges form "bridges" or "shortcuts" between components and are needed due to the fact that v_{dest} (nodes 3, 9) could be connected to some other vertex w with even smaller ID than 0 such that at some point of time node 0 could have to be connected to w. If there were no backward edge $v_{dest} \rightarrow v_{first}$ then there would not be any reducer which would be able to merge 0 and w.

Because of the same arguments, the backward edge $v_{source} \rightarrow v_{first}$ should be actually emitted too. This indeed happens in the case when v_{source} is smaller than one of its adjacent vertices (lines 20 and 21). If v_{source} has the biggest ID among all its adjacent nodes (*optimizedMergeCase*), then this edge can be omitted due to the fact that all adjacent nodes of v_{source} are already reassigned to the vertex with smallest ID and therefore v_{source} will never deal as a bridge node between two components. In Fig. 3(d) case *optimizedMergeCase* is depicted.

The identity mapper and the reducer from Listing 1.1 form one iteration of the CC-MR-algorithm. These jobs have to be iterated as long as there are subcomponents which can be merged. In order to recognize this case, we have to check whether in the last iteration a backward edge was emitted. If this is the case then there are still subcomponents which could be merged and a new iteration has to be started. Otherwise, all components are maximal and the algorithm can stop. The information whether backward edges were created or not is indicated by the global variable *newIterationNeeded*, which can be implemented as a global counter in Hadoop. Setting the value of this variable to e.g. value 1 indicates the boolean value 'true' and value 0 indicates 'false'. This variable is set to true if either case *stdMergeCase* or case *optimizedMergeCase* holds (line 17).

4.2 Example for the Processing of CC-MR

In Fig. 4 we show the processing of CC-MR using the same example that was used in section 3 for the algorithm from [3]. For each iteration, we show the edges that the algorithm emits in this iteration. The vertices that are already connected to the vertex with the smallest ID 0 are marked in the graph for each iteration. In table 1, the output of the single iterations is shown as lists of edges. For each iteration, the all edges that are emitted in this iteration are shown, sorted by their key vertices. Some edges occur repeatedly in the same iteration. This is due to the fact that the same edge can be generated by different reducers.

Table 1. Edges generated by CC-MR for the example graph

iter.	0	1	2	3	4	5	6	7	8
0	0-1	1-0	2-1	3-2	4-3	5-4	6-5	7-6	8-7
		1-2	2-3	3-4	4-5	5-6	6-7	7-8	
1	0-1	1-0	2-0	3-1	4-2	5-3	6-4	7-5	8-6
	0-2	1-3	2-1	3-2	4-3	5-4	6-5	7-6	
			2-4	3-5	4-6	5-7	6-8		
2	0-1 (2x)	1-0	2-0	3-0	4-0	5-1	6-2	7-3	8-4
	0-2	1-0	2-1	3-1	4-2	5-3	6-4		
	0-3	1-2	2-3	3-2	4-3	5-4	6-5		
	0-4	1-5	2-6	3-4	4-5	5-6			
				3-7	4-8				
3	0-1 (3x)	1-0	2-0	3-0	4-0	5-0	6-0	7-0	8-0
	0-2 (4x)	1-0	2-0	3-0	4-0	5-0	6-1		
	0-3 (3x)	1-0	2-0	3-0	4-1	5-1			
	0-4 (2x)	1-3	2-0	3-1	4-2	5-2			
	0-5 (2x)	1-4	2-4						
	0-6	1-6	2-5						
	0-7								
	0-8								

iter.	0	1	2	3	4	5	6	7	8
4	0-1 (5x)	1-0	2-0	3-0	4-0	5-0	6-0		
	0-2 (3x)	1-0	2-0		4-0				
	0-3 (2x)	1-0	2-0						
	0-4 (3x)	1-0							
	0-5 (2x)	1-0							
	0-6 (2x)								
	0-7								
	0-8								
5	0-1								
	0-2								
	0-3								
	0-4								
	0-5								
	0-6								
	0-7								
	0-8								

In the initial graph, for each edge $(u, v) \in E$ both directions, i. e. (u, v) and (v, u) are given. In the first iteration, the reducers mostly insert "two-hop" edges, e.g. the reducer for the vertex 3 connects the vertices 2 and 4. In the second iteration, for example, the reducer for the vertex 2 inserts an edge between the vertices 0 and 4 and the reducer of the vertex 6 inserts an edge between 4 and 8. Thus, vertices are already connected to each other that had a shortest path of 4 in the initial graph. In the third iteration, finally all vertices have direct connections to the node 0. However, to obtain a star graph for this compoment and thus to detect that the final component as already been found, the algorithm still has to delete all edges (v, w) with $v, w \neq 0$. This is done in iteration 4. Though the resulting graph is already the desired star graph, some vertices (i.e. vertex 3) still have backward edges, which are finally removed in a fifth iteration (not depicted here). Overall, our algorithm needs 5 iterations for this example. In comparison, the algorithm from [3] needed 8 iterations in the same example.

4.3 Proof of Correctness

In this section we present a proof for the correctness of CC-MR, i.e. we show that CC-MR correctly detects the connected components of the graph. As presented in the previous section, the idea of our algorithm is to add and delete edges in the graph such that the resulting graph consists of one star graph per component where the center of each star graph is the vertex with the smallest ID from the corresponding component. Thus, in each iteration we have a different set of edges in the graph. Let E_i denote the set of edges that exist after iteration i.

To prove that the resulting graphs of the algorithm really correspond to the connected components, we prove two different steps:

1. An edge (v_1, v_2) is emitted in iteration $i \Rightarrow$ There has already been a path between v_1 and v_2 in iteration $i - 1$.
   (We never add edges between vertices that were not in the same component before.)

2. There exists a path between v_1 and v_2 in iteration $i-1 \Rightarrow$ there exists a path between them in iteration i.
   (We do not disconnect components that existed before).

Steps 1 and 2 together show that although the algorithm adds and removes edges in the graph, the (weak) connected components do never change during the algorithm. Please note that as our input graph is undirected, the connectedness of the components does not depend on the directions of the edges, even though CC-MR sometimes only adds one direction of an edge for optimization reasons. Thus, for the paths we construct in our proof the directions of the edges are neglected. In the following we present the proofs for the single steps:

1. In CC-MR, edges are only added in the reducers. Thus, to add an edge $(v_1, v_2) \in E_i$, the vertices v_1 and v_2 have to occur together in the reducer of some vertex v_{source}. Therefore, for each $v_j, j \in \{1, 2\} : v_j = v_{source}$ or $(v_{source}, v_j) \in E_{i-1}$. Thus, there existed a path between v_1 and v_2 in the iteration $i-1$.
2. It suffices to show: There exists an *edge* $(v_1, v_2) \in E_{i-1} \Rightarrow$ there exists a path between them in iteration i (Because a path between some vertices u and w in E_{i-1} can be reconstructed in E_i by replacing each edge $(v_1, v_2) \in E_{i-1}$ on the path by its corresponding path in E_i):
   Case 1: $v_1 < v_2$ (i.e. (v_1, v_2) is a forward edge):
   (v_1, v_2) is processed in the reducer of $v_{source} = v_1$. Now we can look at the three different cases that can occur in the reducer:
   - *locMaxCase*: (v_1, v_2) is emitted again.
   - *stdMergeCase*: We emit edges (v_{first}, v_2), (v_2, v_{first}) and (v_1, v_{first}), thus there still exists a path between v_1 and v_2.
   - *optimizedMergeCase*: Not possible because $v_1 < v_2$.

   Case 2: $v_1 > v_2$ (i.e. (v_1, v_2) is a backward edge):
   For this case, we can show that in some iteration $i_x \leq i-1$ also the corresponding forward edge (v_2, v_1) was emitted:
   For the backward edge $(v_1, v_2) \in E_{i-1}$, there are two possible scenarios where (v_1, v_2) can have been emitted:
   - $(v_1, v_2) \in E_{i-1}$ can have been emitted by the reducer of v_1 in the case *stdMergeCase*. In this case, the edge has already existed in the previous iteration , i.e. $(v_1, v_2) \in E_{i-2}$, else the vertex v_2 would not be processed in the reducer of v_1.
   - $(v_1, v_2) \in E_{i-1}$ can have been emitted by the reducer of a vertex v_{source} with $v_{source} \neq v_1$ and $v_{source} \neq v_2$ in the case *stdMergeCase* or *optimizedMergeCase*. In this case, also the forward edge (v_2, v_1) has been emitted.
   - $i-1 = 0$, i.e. the edge (v_1, v_2) already existed in the initial graph. Then, by definition of the original (undirected) graph, also the forward edge (v_2, v_1) existed.

Thus, we know that for each backward edge (v_1, v_2), the corresponding forward edge (v_2, v_1) exists or has existed in an earlier iteration. In case 1 it was already shown that the path between v_2 and v_1 is preserved in the following iterations. □

4.4 Dealing with Large Components

In CC-MR a single reducer processes a complete component, which can result in high workloads of a single reducer for very large components. Now we briefly present a solution which distributes the calculations for large components over multiple reducers and that way balances the workload. Consider the example in Fig. 5, in which vertex 7 is a center of a component with too many elements. Assume that we have this information from a previous iteration and we want to distribute the computations on three reducers. For that, in the map-phase

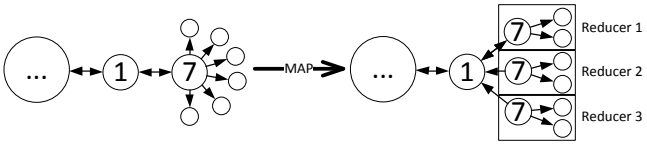


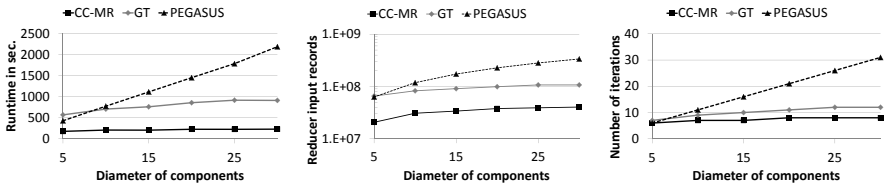
Fig. 5. Example for workload balancing for big components

each forward edge $(7, x)$ is augmented by a hash value to $((7, hash(x)), x)$ which is then used in the partitioner in order to distribute edges $(7, \cdot)$ to different reducers. In the example a small circle represents such a vertex x, which is then sent to one of the three reducers. For backward edges (in the example $(7, 1)$), a set of edges $\{((7, hash(i)), 1) | i = 1, 2, 3\}$ is produced, which guarantees that each vertex 7 in each of the three reducers has a backward edge to 1 and can reassign its neighbors to 1 in the reduce phase. All other edges whose source vertices do not have too many neighbors are not augmented by a hash value and therefore are processed as in the original algorithm.

The reduce-phase remains as in the original algorithm, with the difference that for each vertex the number of neighbors in E_i is determined and for vertices with too many neighbors the value is stored in the distributed cache for the next iteration. This simple strategy produces almost no additional overhead but achieves a very good balancing of the workload as will be shown in the experiments.

5 Experiments

In this section we present the experimental evaluation of the CC-MR approach, comparing it to the approach from [3] (denoted as 'GT' here) and to the approach from the Pegasus system [7]. All experiments were performed on a cluster running



(a) Component diameter vs. runtime (b) Component diameter vs. number of input records (c) Component diameter vs. number of iterations

Fig. 6. Performance for varying component diameters

Hadoop 0.20.2 and consisting of 14 nodes with 8 cores each that are connected via a 1 Gbit network. Each of the nodes has 16 Gb RAM. For each experiment the number of distance computations and the runtime is measured[3].

5.1 Scalability on Synthetic Data

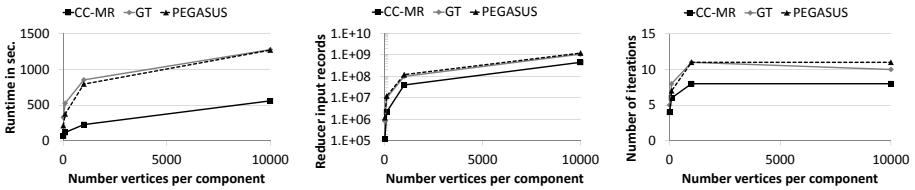
In this section we evaluate CC-MR with respect to different properties of the input graphs. Therefore, we use synthetic datasets such that in each experiment, one property of the generated graphs changes while the others remain stable.

Performance for Varying Component Diameters. In this experiment we vary the diameter of the connected components in our synthetic datasets. Each dataset consists of one million vertices and is divided into 1000 connected components with 1000 vertices each. The diameter of the generated components is varied from 10 to 30 in this experiment.

In Fig. 6(a), the runtime of the algorithms is depicted. For all algorithms the runtime increases for higher diameters. However, the runtime of CC-MR is always significantly lower (at least a factor of two) than that of GT and Pegasus and scales better for increasing diameters. In Fig. 6(c) we show the number of iterations needed by the algorithms to find the connected components. As expected, for higher diameters the number of iterations increases for all algorithms. For CC-MR, the number of iterations is always lower than for GT and Pegasus. In Fig. 6(b), we depict for each algorithm the number of input records (summed over all iterations) that are processed, i.e. the number of records that are communicated between the iterations. For all algorithms, this number increases with increasing diameter. The number of records for CC-MR is significantly lower (at least by a factor of two) than that of GT and Pegasus, due to the fact that they perform more iterations than CC-MR and GT needs to perform 3 MapReduce jobs per iteration.

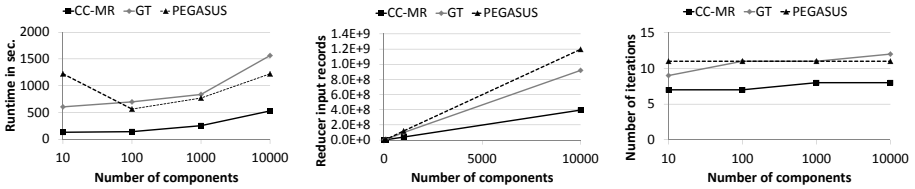
Performance for Varying Component Sizes. In this experiment we vary the number of vertices per component in the generated graphs and examine its influence on the runtime and the number generated records in reducers. The created datasets consist of 1000 components with 15, 150, 1500 and 15000 edges and

[3] The source code of the CC-MR-algorithm and the used datasets can be found at: http://dme.rwth-aachen.de/en/ccmr



(a) Component size vs. runtime (b) Component size vs. number of input records (c) Component size vs. number of iterations

Fig. 7. Performance for varying component sizes



(a) Number of components vs. runtime (b) Number of components vs. number of input records (c) Number of components vs. number of iterations

Fig. 8. Performance for varying numbers of components

10, 100, 1000, 10000 vertices, respectively. The figures 7(a), 7(b) and 7(c) depict the results for the runtime (in sec.), the number of records, and the iteration number respectively. As expected, all of these values increase with growing size of the components. The runtime of the CC-MR algorithm remains smaller (up to a factor of 2) then the runtimes of GT and Pegasus for each component size.

Performance for Varying Numbers of Components. This experiment shows the dependency between the number of components in a graph, the runtime, the number of input records and the number of iterations. The synthetic datasets consist of 10, 100, 1000, 10000 components each with 1000 vertices and 1500 edges. The figures 8(a), 8(b) and 8(c) depict the runtime, the processed number of input records and the number of iterations. Similar to previous results, CC-MR has a much lower runtime and produces in the worst case (10 components) 15% and in the best case (10000 components) over 55% less input records compared to the other approaches. Furthermore, CC-MR outperforms both competitors in terms of the number of performed iterations.

5.2 Real-world Data

We use three real-world datasets to evaluate CC-MR: a web graph (Web-google) and two collaboration networks (IMDB, DBLP). The Web-google dataset can be found at `snap.stanford.edu/data/web-Google.html` and consists of 875713 nodes, 5105039 edges and 2746 connected components. In this web graph, the vertices represent web pages and the edges represent hyperlinks between them.

Table 2. Results of CC-MR, GT and PEGASUS

	CC-MR			GT			PEGASUS		
	web-google	imdb	dblp	web-google	imdb	dblp	web-google	imdb	dblp
Runtime (sec.)	535	1055	472	3567	3033	4385	4847	2834	3693
#iters	8	6	7	10	6	12	16	6	15
#input rec. ($\cdot 10^6$)	29	179	27	102	564	210	292	299	108

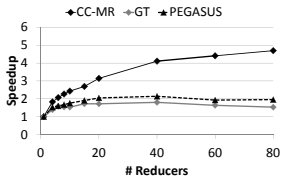


Fig. 9. Speedup for varying number of reduce nodes

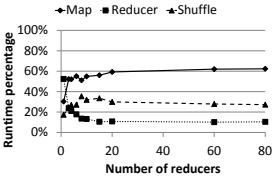


Fig. 10. CC-MR-Runtime distribution among map, shuffle and reduce phases

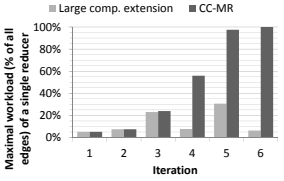


Fig. 11. Maximal reducer workload for processing large components

In the IMDB dataset (176540 nodes; 19992184 edges; 16 comps.), actors are represented by vertices, edges between actors indicate that the actors worked together in some movie. The data was extracted from the IMDB movie database (`imdb.com`). In the DBLP dataset (generated using the data from `dblp.org`, 553797 nodes; 1677487 edges; 24725 comps.) each vertex corresponds to an author, while each edge represents a collaboration between two authors.

The results of the comparison are given in Table 2. CC-MR clearly outperforms GT and PEGASUS in terms of the number of iterations as well as in the runtime and the communication (i.e the number of input records).

Figure 9 depicts the scalability results of the evaluated algorithms. The CC-MR-algorithm shows a speedup more than twice as high compared with competing approaches. As Figure 10 shows, the reduce time of our approach decreases very fast with growing number of reducers. The moderate overall speedup of 4.7 with 80 reducers puts down to the fact that the map phase does not depend on the number of used reducers and therefore the speedup of calculations is limited by I/O speed. The significant communication reduction of our approach is therefore a very big advantage in comparison to the competing approaches.

5.3 Load Balancing for Large Components

Fig. 11 shows the load balancing properties of our large component extension (cf. Section 4.4) on the IMDB dataset using 20 reducers and threshold for the maximal size of a component set to 1% of the number of edges in an iteration. This dataset contains 16 components, 15 small ones and one, C_{max}, containing more than 99% of the vertices. In the first three iterations both algorithms perform similarly well and distribute the workload almost evenly among all reducers. In iteration 4, however, the reducer R_{max} of the CC-MR responsible for C_{max}

already processes about 50% of all edges remaining in the iteration, as more and more vertices are assigned to C_{max}. This trend goes on until in the last iteration almost all vertices of the graph are processed by R_{max}. In contrast, our extended algorithm is able to balance the workload after each iteration when a disbalance occurs, as it is the case in the iterations 3 and 5.

6 Conclusion

In this paper, we propose the parallel algorithm CC-MR for the detection of the connected components of a graph. The algorithm is built on top of the MapReduce programming model. CC-MR effectively manipulates the graph structure to reduce the number of needed iterations and thus to find the connected components more quickly. Furthermore, we propose an extension to CC-MR to deal with heterogeneous component sizes. Apart from the description of the algorithm, we also provide a proof for the correctness of CC-MR. The performance of CC-MR is evaluated on synthetic and real-world data in the experimental section. The experiments show that CC-MR constantly outperforms the state-of-the-art approaches.

References

1. Bus, L., Tvrdík, P.: A Parallel Algorithm for Connected Components on Distributed Memory Machines. In: Cotronis, Y., Dongarra, J. (eds.) PVM/MPI 2001. LNCS, vol. 2131, pp. 280–287. Springer, Heidelberg (2001)
2. Chin, F.Y.L., Lam, J., Chen, I.-N.: Efficient parallel algorithms for some graph problems. Commun. ACM 25(9), 659–665 (1982)
3. Cohen, J.: Graph twiddling in a MapReduce world. Computing in Science and Engineering 11(4), 29–41 (2009)
4. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: OSDI, pp. 137–150 (2004)
5. Greiner, J.: A comparison of parallel algorithms for connected components. In: SPAA, pp. 16–25 (1994)
6. Hirschberg, D.S., Chandra, A.K., Sarwate, D.V.: Computing connected components on parallel computers. Commun. ACM 22(8), 461–464 (1979)
7. Kang, U., Tsourakakis, C.E., Faloutsos, C.: Pegasus: A peta-scale graph mining system. In: ICDM, pp. 229–238 (2009)
8. Krishnamurthy, A., Lumetta, S., Culler, D., Yelick, K.: Connected components on distributed memory machines. DIMACS Implementation Challenge 30, 1 (1997)
9. Lattanzi, S., Moseley, B., Suri, S., Vassilvitskii, S.: Filtering: a method for solving graph problems in mapreduce. In: SPAA, pp. 85–94 (2011)
10. Rastogi, V., Machanavajjhala, A., Chitnis, L., Sarma, A.D.: Finding connected components on map-reduce in logarithmic rounds. Computing Research Repository (CoRR), abs/1203.5387 (2012)
11. Shiloach, Y., Vishkin, U.: An o(log n) parallel connectivity algorithm. J. Algorithms 3(1), 57–67 (1982)
12. Wu, B., Du, Y.: Cloud-based connected component algorithm. In: Artificial Intelligence and Computational Intelligence (AICI), vol. 3, pp. 122–126 (2010)

Fast Near Neighbor Search in High-Dimensional Binary Data

Anshumali Shrivastava and Ping Li

Cornell University, Ithaca NY 14853, USA

Abstract. Numerous applications in search, databases, machine learning, and computer vision, can benefit from efficient algorithms for near neighbor search. This paper proposes a simple framework for *fast near neighbor search in high-dimensional binary data*, which are common in practice (e.g., text). We develop a very simple and effective strategy for sub-linear time near neighbor search, by creating hash tables directly using the bits generated by b-bit minwise hashing. The advantages of our method are demonstrated through thorough comparisons with two strong baselines: *spectral hashing* and *sign (1-bit) random projections*.

1 Introduction

As a fundamental problem, the task of *near neighbor search* is to identify a set of data points which are "most similar" to a query data point. Efficient algorithms for near neighbor search have numerous applications in the context of search, databases, machine learning, recommending systems, computer vision, etc.

Consider a data matrix $\mathbf{X} \in \mathbb{R}^{n \times D}$, i.e., n samples in D dimensions. In modern applications, both n and D can be large, e.g., billions or even larger [1]. Intuitively, near neighbor search may be accomplished by two simple strategies. The first strategy is to pre-compute and store all pairwise similarities at $O(n^2)$ space, which is only feasible for small number of samples (e.g., $n < 10^5$).

The second simple strategy is to scan all n data points and compute similarities on the fly, which however also encounters difficulties: (i) The data matrix $\mathbf{X}$ itself may be too large for the memory. (ii) Computing similarities on the fly can be too time-consuming when the dimensionality D is high. (iii) The cost of scanning all n data points is prohibitive and may not meet the demand in user-facing applications (e.g., search). (iv) Parallelizing linear scans will not be energy-efficient if a significant portion of the computations is not needed.

Our proposed (simple) solution is built on the recent work of b-bit minwise hashing [2,3] and is specifically designed for binary high-dimensional data.

1.1 Binary, Ultra-High Dimensional Data

For example, consider a Web-scale term-doc matrix $\mathbf{X} \in \mathbb{R}^{n \times D}$ with each row representing one Web page. Then roughly $n = O(10^{10})$. Assuming 10^5 common English words, then the dimensionality $D = O(10^5)$ using the uni-gram model

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 474–489, 2012.

and $D = O(10^{10})$ using the bi-gram model. Certain industry applications used 5-grams [4,5,6] (i.e., $D = O(10^{25})$ is conceptually possible). Usually, when using 3- to 5-grams, most of the grams only occur at most once in each document. It is thus common to utilize only binary data when using n-grams.

1.2 *b*-Bit Minwise Hashing

Minwise hashing [5] is a standard technique for efficiently computing set similarities in the context of search. The method mainly focuses on binary (0/1) data, which can be viewed as sets. Consider two sets $S_1, S_2 \subseteq \Omega = \{0, 1, 2, ..., D-1\}$, the method applies a random permutation $\pi : \Omega \to \Omega$ on S_1 and S_2 and utilizes

$$\mathbf{Pr}\left(\min(\pi(S_1)) = \min(\pi(S_2))\right) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = R \tag{1}$$

to estimate R, the resemblance between S_1 and S_2. A prior common practice was to store each hashed value, e.g., $\min(\pi(S_1))$, using 64 bits [6], which can lead to prohibitive storage and computational costs in certain industrial applications [7]. b-bit minwise hashing [2] is a simple solution by storing only the lowest b bits of each hashed value. For convenience, we define

$$z_j = \min(\pi(S_j)), \qquad z_j^{(b)} = \text{the lowest } b \text{ bits of } z_j.$$

Assuming D is large, [2] derived a new collision probability:

$$P_b(R) = \mathbf{Pr}\left(z_1^{(b)} = z_2^{(b)}\right) = C_{1,b} + (1 - C_{2,b})\, R \tag{2}$$

$$r_1 = \frac{f_1}{D}, \quad r_2 = \frac{f_2}{D}, \quad f_1 = |S_1|, \quad f_2 = |S_2|$$

$$C_{1,b} = A_{1,b}\frac{r_2}{r_1 + r_2} + A_{2,b}\frac{r_1}{r_1 + r_2}, \qquad C_{2,b} = A_{1,b}\frac{r_1}{r_1 + r_2} + A_{2,b}\frac{r_2}{r_1 + r_2},$$

$$A_{1,b} = \frac{r_1\left[1 - r_1\right]^{2^b - 1}}{1 - \left[1 - r_1\right]^{2^b}}, \qquad A_{2,b} = \frac{r_2\left[1 - r_2\right]^{2^b - 1}}{1 - \left[1 - r_2\right]^{2^b}}.$$

This result suggests an unbiased estimator of R from k permutations π_1, ..., π_k:

$$\hat{R}_b = \frac{\hat{P}_b - C_{1,b}}{1 - C_{2,b}}, \qquad \hat{P}_b = \frac{1}{k}\sum_{j=1}^{k} 1\{z_{1,\pi_j}^{(b)} = z_{2,\pi_j}^{(b)}\} \tag{3}$$

whose variance would be

$$\mathrm{Var}\left(\hat{R}_b\right) = \frac{1}{k}\frac{\left[C_{1,b} + (1 - C_{2,b})R\right]\left[1 - C_{1,b} - (1 - C_{2,b})R\right]}{\left[1 - C_{2,b}\right]^2} \tag{4}$$

The advantage of b-bit minwise hashing can be demonstrated through the "variance-space" trade-off: $\mathrm{Var}\left(\hat{R}_b\right) \times b$. Basically, when the data are highly similar, a small b (e.g., 1 or 2) may be good enough. However, when the data are not very similar, b can not be too small.

1.3 Our Proposal for Sub-linear Time Near Neighbor Search

Our proposed method is simple, by directly using the bits generated from b-bit minwise hashing to build hash tables, which allow us to search near neighbors in sub-linear time (i.e., no need to scan all data points).

Specifically, we hash the data points using k random permutations and store each hash value using b bits (e.g., $b \leq 4$). For each data point, we concatenate the resultant $B = b \times k$ bits as a *signature*. The size of the space is $2^B = 2^{b \times k}$, which is not too large for small b and k (e.g., $bk = 16$). This way, we create a table of 2^B buckets, numbered from 0 to $2^B - 1$; and each bucket stores the pointers of the data points whose signatures match the bucket number. In the testing phrase, we apply the same k permutations to a query data point to generate a bk-bit signature and only search data points in the corresponding bucket.

Of course, using only one hash table will likely miss many true near neighbors. As a remedy, we generate (using independent random permutations) L hash tables; and the query result is the union of the data points retrieved in L tables.

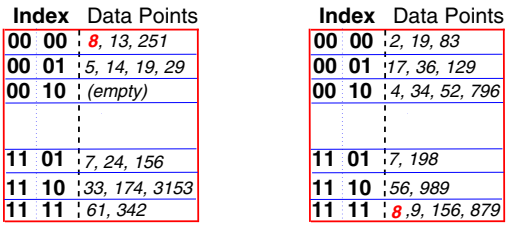


Fig. 1. An example of hash tables, with $b = 2$, $k = 2$, and $L = 2$

In the example in Figure 1, we choose $b = 2$ bits and $k = 2$ permutations, i.e., one hash table has 2^4 buckets. Given n data points, we apply $k = 2$ permutations and store $b = 2$ bits of each hashed value to generate n (4-bit) signatures. Consider data point 8. After $k = 2$ permutations, the lowest b-bits of the hashed values are 00 and 00. Therefore, its signature is 0000 in binary and hence we place a pointer to data point 8 in bucket number 0 (as in the left panel of Figure 1).

In this example, we choose to build $L = 2$ tables. Thus we apply another $k = 2$ permutations and place the n data points to the second table (as in the right panel of Figure 1) according to their signatures. This time, the signature of data point 8 becomes 1111 in binary and hence we place it in the last bucket.

Suppose in the testing phrase, the two (4-bit) signatures of a new data point are 0000 and 1111, respectively. We then only search the near neighbors in the set $\{8, 9, 13, 156, 251, 879\}$, which is much smaller than the set of n data points.

2 Other Methods for Efficient Near Neighbor Search

Developing efficient algorithms for finding near neighbors has been an active research topic since the early days of modern computing. For example, K-D trees [12] and variants often work reasonably well in very low-dimensional data.

Our technique can be viewed as an instance of *Locality Sensitive Hashing (LSH)* [13,14,15], which represents a very general family of algorithms for near neighbor search. The performance of any LSH scheme depends on the underlying algorithm. Our idea of directly using the bits generated from b-bit miniwise hashing to build hash tables is novel and requires own analysis.

The effectiveness of our proposed algorithm can be demonstrated through thorough comparisons with strong baselines. In this paper, we focus on *spectral hashing (SH)* [8] and the LSH based on *sign random projections* [9,10,11].

2.1 Centered and Noncentered Spectral Hashing (SH-C, SH-NC)

Spectral hashing (SH) [8] is a representative example of "learning-based hashing" algorithms, which typically require a (very) expensive training step. It appears that more recent learning-based hashing algorithms, e.g., [16,17] have not shown a definite advantage over SH. Moreover, other learning-based search algorithms are often much more complex than SH. Thus, to ensure our comparison study is fair and repeatable, we focus on SH.

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times D}$, SH first computes the top eigenvectors of the sample covariance matrix and maps the data according to the top eigenvectors. The mapped data are then thresholded to be binary (0/1), which are the hash code bits for near neighbor search. Clearly, for massive high-dimensional data, SH is prohibitively memory-intensive and time-consuming. Also, storing these eigenvectors (for testing new data) requires excessive disk space when D is large.

We made two modifications to the original SH implementation [8]. Here, we quote from their Matlab code [8] to illustrate the major computational cost:

```
[pc, l] = eigs(cov(X), npca);   X = X * pc;
```

Our first modification is to replace the eigen-decomposition by SVD, which avoids materializing the covariance matrix (of size $D \times D$). That is, we first remove the mean from $\mathbf{X}$ (called "centering") and then apply Matlab "svds" (instead of "eigs") on the centered $\mathbf{X}$. This modification can substantially reduce the memory consumption without altering the results.

The centering step (or directly using "eigs"), however, can be disastrous because after centering the data are no longer sparse. For example, with centering, training merely 4000 data points in about 16 million dimensions (i.e., the *Webspam* dataset) took 2 days in a workstation with 96GB memory, to obtain 192-bit hash codes. Storing those 192 eigenvectors consumed 24GB disk space after compression (using the "-v7.3" save option in Matlab).

In order to make reliable comparisons with SH, we implemented both centered version (SH-C) and noncentered version (SH-NC). Since we focus on binary data

in this study, it is not clear if centering is at all necessary. In fact, our experiments will show that SH-NC often perform similarly as SH-C.

Even with the above two modifications, SH-NC is still very expensive. For example, it took over one day for training 35,000 data points of the *Webspam* dataset to produce 256-bit hash code. The prohibitive cost for storing the eigenvectors remains the same as SH-C (about 32GB for 256 bits).

Once the hash code has been generated, searching for near neighbors amounts to finding data points whose hash codes are closest (in *hamming distance*) to the hash code of the query point [8]. Strictly speaking, there is no proof that one can build hash tables using the bits of SH in the sense of LSH. Therefore, to ensure that our comparisons are fair and repeatable, we only experimentally compare the code quality of SH with b-bit minwise hashing, in Section 3.

2.2 Sign Random Projections (SRP)

The method of random projections utilizes a random matrix $P \in \mathbb{R}^{D\times k}$ whose entries are i.i.d. normal, i.e., $P_{ij} \sim N(0,1)$. Consider two sets S_1, S_2. One first generates two projected vectors $v_1, v_2 \in \mathbb{R}^k$: $v_{1j} = \sum_{i\in S_1} P_{ij}$, $\quad v_{2j} = \sum_{i\in S_2} P_{ij}$, and then estimates the size of intersection $a = |S_1 \cap S_1|$ by $\frac{1}{k}\sum_{j=1}^{k} v_{1j}v_{2j}$.

It turns out that this method is not accurate as shown in [3]. Interestingly, using only the signs of the projected data can be much more accurate (in terms of variance per bit). Basically, the method of *sign random projections* estimates the similarity using the following collision probability:

$$\mathbf{Pr}\left(\text{sign}(v_{1,j}) = \text{sign}(v_{2,j})\right) = 1 - \frac{\theta}{\pi}, \qquad j = 1, 2, ..., k, \tag{5}$$

where $\theta = \cos^{-1}\left(\frac{a}{\sqrt{f_1 f_2}}\right)$ is the angle. This formula was presented in [9] and popularized by [10]. The variance was analyzed and compared in [11].

We will first compare SRP with b-bit minwise hashing in terms of hash code quality. We will then build hash tables to compare the performance in sub-linear time near neighbor search. See Appendix A for the variance-space comparisons.

3 Comparing Hash Code Quality

We tested three algorithms (b-bit, SH, SRP) on two binary datasets: *Webspam* and *EM30k*. The *Webspam* dataset was used in [3], which also demonstrated that using the binary-quantized version did not result in loss of classification accuracy. For our experiments, we sampled $n = 70,000$ examples from the original dataset. The dimensionality is $D = 16,609,143$.

The *EM30k* dataset was used in [18] to demonstrate the effectiveness of image feature expansions. We sampled $n = 30,000$ examples from the original dataset. The dimensionality is $D = 34,950,038$.

3.1 The Evaluation Procedure

We evaluate the algorithms in terms of the **precision-recall** curves. Basically, for each data point, we sort all other data points in the dataset in descending (estimated) similarities using the hash code of length B. We walk down the list (up to 1000 data points) to retrieve the "top T" data points, which are most similar (in terms of the original similarities) to that query point. We choose $T = 5$, $T = 10$, $T = 20$, and $T = 50$. The precision and recall are defined as:

$$\text{Precision} = \frac{\#\text{ True Positive}}{\#\text{Retrieved}}, \qquad \text{Recall} = \frac{\#\text{ True Positive}}{T} \tag{6}$$

We vary # retrieved data points from 1 to 1000 spaced at 1, to obtain continuous precision-recall curves. The final results are averaged over all the test data points.

3.2 Experimental Results on Webspam (4000)

We first experimented with 4000 data points from the *Webspam* dataset, which is small enough so that we could train the centered version of spectral hashing (i.e., SH-C). On a workstation with 96 GB memory, SH-C took about 2 days and used about 90GB memory at the peak, to produce 192-bit hash code.

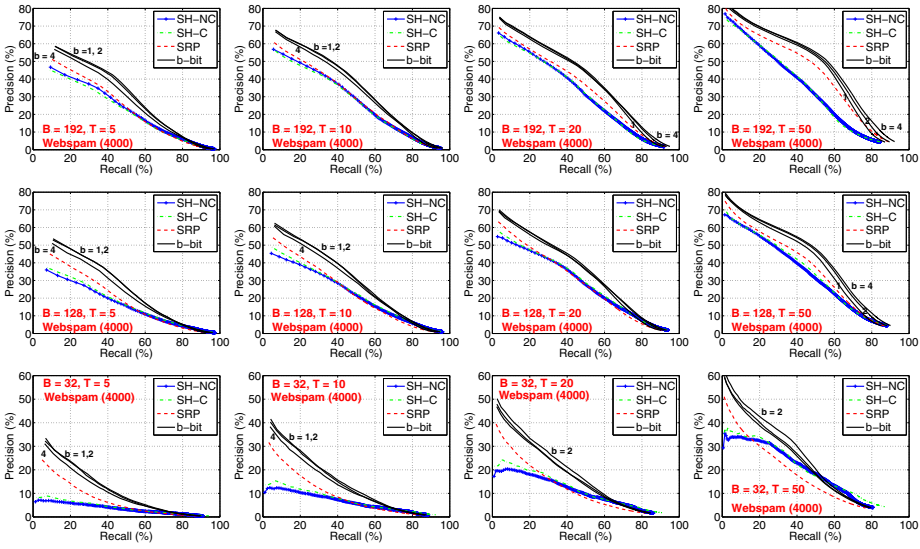


Fig. 2. Precision-recall curves (the higher the better) for all four methods (SRP, b-bit, SH-C, and SH-NC) on a small subset (4000 data points) of the *Webspam* dataset. The task is to retrieve the top T near neighbors (for $T = 5$, 10, 20, 50). B is the bit length.

Figure 2 presents the results of b-bit hashing, SH-C, SH-NC, and SRP in terms of the precision-recall curves (the higher the better), for $B = 192$, 128, and 32

bits. Basically, for b-bit hashing, we choose $b = 1, 2, 4$ and k so that $b \times k = B$. For example, if $B = 192$ and $b = 2$, then $k = 96$. As analyzed in [2], for a pair of data points which are very similar, then using smaller b will outperform using larger b in terms of the variance-space tradeoff. Thus, it is not surprising if $b = 1$ or 2 shows better performance than $b = 4$ for this dataset.

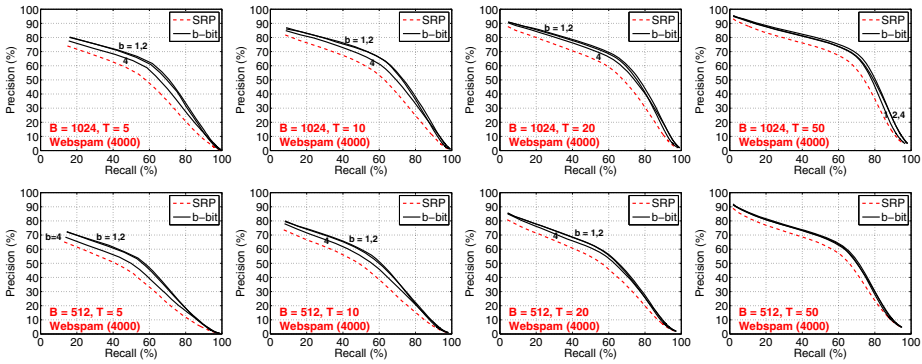


Fig. 3. Precision-recall curves for SRP and b-bit hashing on 4000 data points of the *Webspam* dataset using 1024-bit and 512-bit codes, for which we could not run SH

Figure 3 compares b-bit hashing with SRP with much longer hash code (1024 bits and 512 bits). These two figures demonstrate that:

- SH-C and SH-NC perform very similarly in this case, while SH-NC is substantially less expensive (several hours as opposed to 2 days).
- SRP is better than SH and is noticeably worse than b-bit hashing for all b.

We need to clarify how we obtained the gold-standard list for each method. For b-bit hashing, we used the original resemblances. For SRP, we used the original cosines. For SH, following [8] we used the original Euclidian distances.

3.3 Experimental Results on Webspam (35000)

Based on 35000 (which are more reliable than 4000) data points of the *Webspam* dataset, Figure 4 again illustrates that SRP is better than SH-NC and is worse than b-bit hashing. Note that we can not train SH-C on 35000 data points. We limited the SH bit length to 256 because 256 eigenvectors already occupied 32GB disk space after compression. On the other hand, we can use much longer code lengths for the two inexpensive methods, SRP and b-bit hashing. As shown in Figure 5, for 512 bits and 1024 bits, b-bit hashing still outperformed SRP.

3.4 EM30k (15000)

For this dataset, as the dimensionality is so high, it is difficult to train SH at a meaningful scale. Therefore, we only compare SRP with b-bit hashing in Figure 6, which clearly demonstrates the advantage of b-bit minwise hashing.

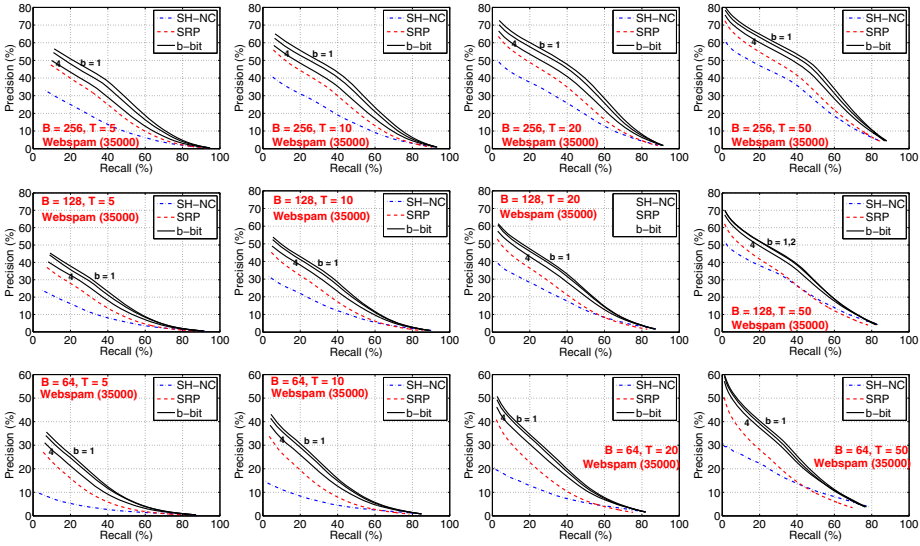


Fig. 4. Precision-recall curves for three methods (SRP, b-bit, and SH-NC) on 35000 data points of the *Webspam* dataset. Again, b-bit outperformed SH and SRP.

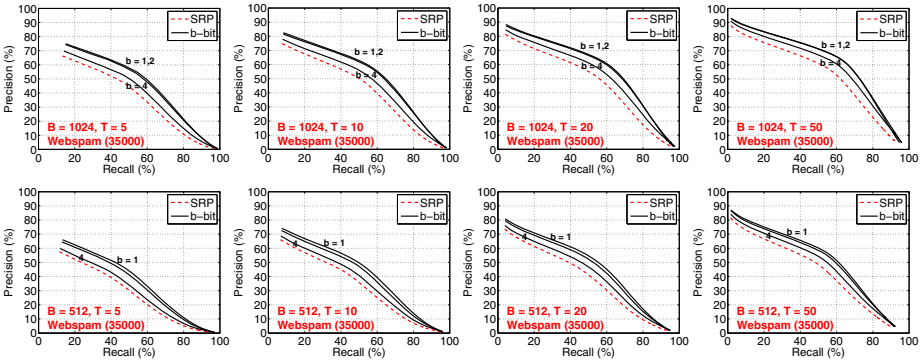


Fig. 5. Precision-recall curves for SRP and b-bit minwise hashing on 35000 data points of the *Webspam* dataset, for longer code lengths (1024 bits and 512 bits)

4 Sub-linear Time Near Neighbor Search

We have presented our simple strategy in Section 1.3 and Figure 1. Basically, we apply k permutations to generate one hash table. For each permutation, we store each hashed data using only b bits and concatenate k b-bit strings to form a *signature*. The data point (in fact, only its pointer) is placed in a table of 2^B buckets ($B = b \times k$). We generate L such hash tables using independent permutations. In the testing phrase, given a query data point, we apply the same

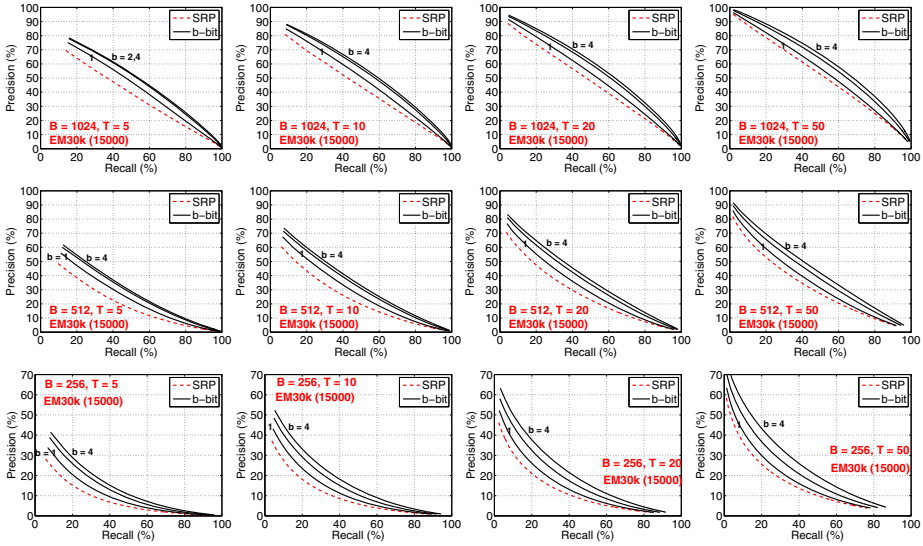


Fig. 6. Precision-recall curves for SRP and b-bit minwise hashing on 15000 data points of the *EM30k* dataset

random permutations to generate signatures and only search for data points (called the *candidate set*) in the corresponding buckets.

In the next step, there are many possible ways of selecting near neighbors from the candidate set. For example, suppose we know the exact nearest neighbor has a resemblance R_0 and our goal is to retrieve T points whose resemblances to the query point are $\geq cR_0$ ($c < 1$). Then we just need to keep scanning the data points in the candidate set until we encounter T such data points, assuming that we are able to compute the exact similarities. In reality, however, we often do not know the desired threshold R_0, nor do we have a clear choice of c. Also, we usually can not afford to compute the exact similarities.

To make our comparisons easy and fair, we simply re-rank all the retrieved data points and compute the precision-recall curves by walking down the list of data points (up to 1000) sorted by descending order of similarities. For simplicity, to re-rank the data points in the candidate set, we use the estimated similarities from $k \times L$ permutations and b bits per hashed value.

4.1 Theoretical Analysis

Collision Probability. Eq. (2) presents the basic collision probability $P_b(R)$. After the hash tables have been constructed with parameters b, k, L, we can easily write down the overall collision probability (a commonly used measure):

$$P_{b,k,L}(R) = 1 - \left(1 - P_b^k(R)\right)^L \tag{7}$$

which is the probability at which a data point with similarity R will match the signature of the query data point at least in one of the L hash tables.

For simplicity, in this section we will always assume that the data are sparse, i.e., $r_1 \to 0$, $r_2 \to 0$ in (2) which leads to convenient simplification of (2):

$$P_b(R) = \frac{1}{2^b} + \left(1 - \frac{1}{2^b}\right) R. \tag{8}$$

Required Number of Tables L. Suppose we require $P_{b,k,L}(R) > 1 - \delta$, then the number of hash tables (denoted by L) should be

$$L \geq \frac{\log 1/\delta}{\log\left(\frac{1}{1-P_b^k(R)}\right)} \tag{9}$$

which can be satisfied by a combination of b and k. The optimal choice depends on the threshold level R, which is often unfortunately unknown in practice.

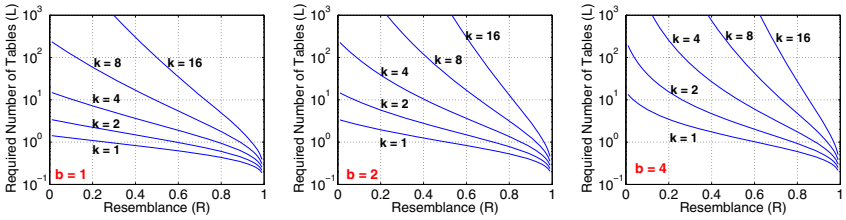


Fig. 7. Required number of tables (L) as in (9), without the $\log 1/\delta$ term. The numbers in the plots should multiply by $\log 1/\delta$, which is about 3 when $\delta = 0.05$.

Number of Retrieved Points before Re-ranking. The expected number of total retrieved points (before re-ranking) is an integral, which involves the data distribution. For simplicity, by assuming a uniform distribution, the fraction of the data points retrieved before the re-ranking step would be (See Appendix B):

$$\int_0^1 P_{b,k,L}(tR)dt = 1 - \sum_{i=0}^{L} \binom{L}{i} (-1)^i \frac{1}{2^{bki}} \frac{1}{(2^b-1)R} \frac{\left((2^b-1)R+1\right)^{ki+1} - 1}{ki+1} \tag{10}$$

Figure 8 plots (10) to illustrate that the value is small for a range of parameters.

Threshold Analysis. To better view the threshold, one commonly used strategy is to examine the point R_0 where the 2nd derivative is zero (i.e., the inflection point of $P_{b,k,L}(R)$): $\left.\frac{\partial^2 P_{b,k,L}}{\partial R^2}\right|_{R_0} = 0$, which turns out to be:

$$R_0 = \frac{\left(\frac{k-1}{Lk-1}\right)^{1/k} - \frac{1}{2^b}}{1 - \frac{1}{2^b}} \tag{11}$$

Figure 9 plots (11). For example, suppose we fix $L = 100$ and $B = b \times k = 16$. If we use $b = 4$, then $R_0 \approx 0.52$. If we use $b = 2$, then $R_0 \approx 0.4$. In other words, a larger b is preferred if we expect that the near neighbors have low similarities.

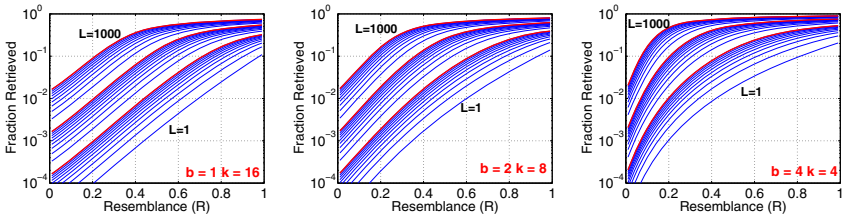


Fig. 8. Numerical values for (10), the fraction of retrieved points

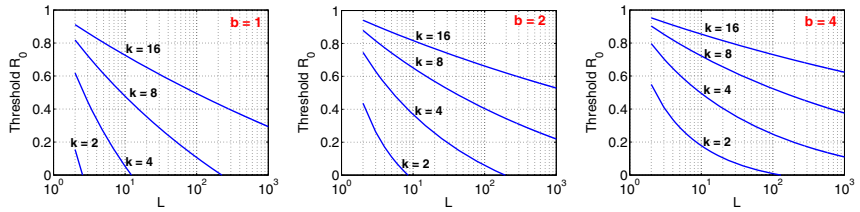


Fig. 9. The threshold R_0 computed by (11), i.e., inflection point of $P_{b,k,L}(R)$

4.2 Experimental Results on the Webspam Dataset

We use 35,000 data points to build hash tables and another 35,000 data points for testing. We build hash tables from both b-bit minwise hashing and sign random projections, to conduct shoulder-by-shoulder comparisons.

Figure 10 plots the fractions of retrieved data points before re-ranking. b-bit hashing with $b = 1$ or 2 retrieves similar numbers of data points. This means, if we also see that the b-bit hashing (with $b = 1$ or 2) has better precision-recall curves than SRP, we know that b-bit hashing is definitely better.

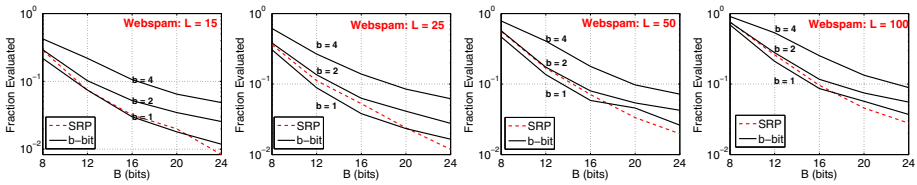


Fig. 10. Fractions of retrieved data points (before re-ranking) on the *Webspam* dataset

Figures 11 and 12 plot the precision-recall curves for $L = 100$ and 50 tables, respectively, demonstrating the advantage of b-bit minwise hashing over SRP.

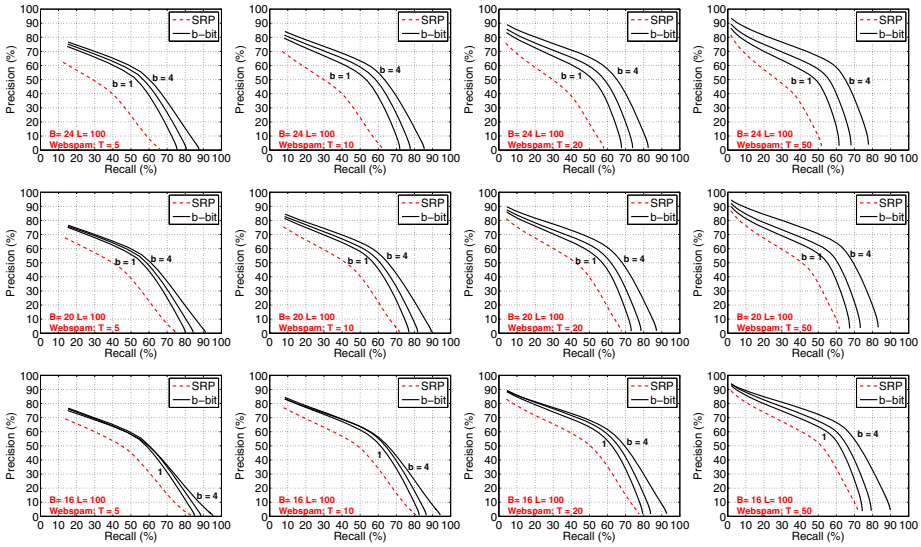


Fig. 11. Precision-recall curves for SRP and b-bit minwise hashing on the *Webspam* dataset using $L = 100$ tables, for top $T = 5$, 10, 20, and $T = 50$ near neighbors

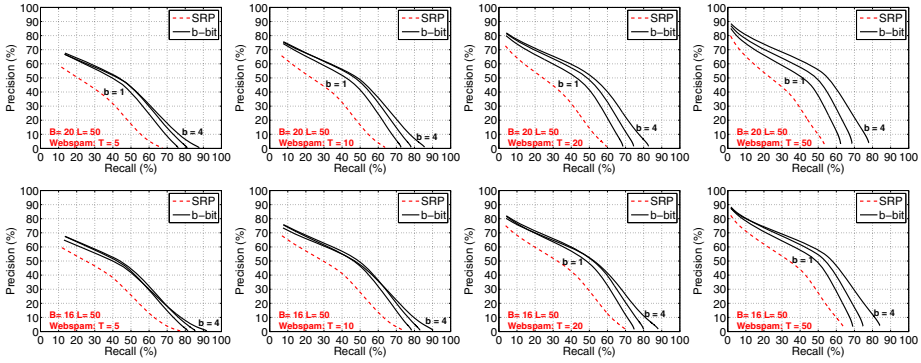


Fig. 12. Precision-Recall curves for SRP and b-bit minwise hashing on the *Webspam* dataset, using $L = 50$ tables

4.3 Experimental Results on EM30k Dataset

For this dataset, we choose 5000 data points (out of 30000) as the query points and use the rest 25000 points for building hash tables.

Figure 13 plots the # retrieved data points before the re-ranking step. We can see that b-bit hashing with $b = 2$ retrieves similar numbers of data points. Again, this means, if we also see that the b-bit hashing (with $b = 1$ or 2) has better precision-recall curves than SRP, then b-bit hashing is certainly better.

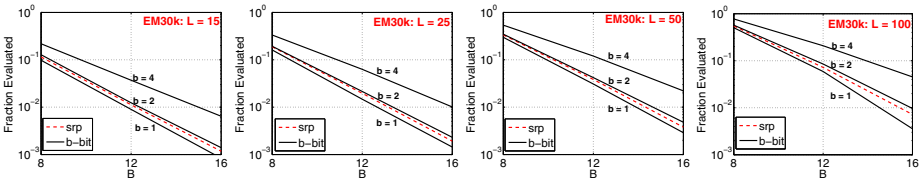


Fig. 13. Fractions of retrieved data points (before re-ranking) on the *EM30k* dataset

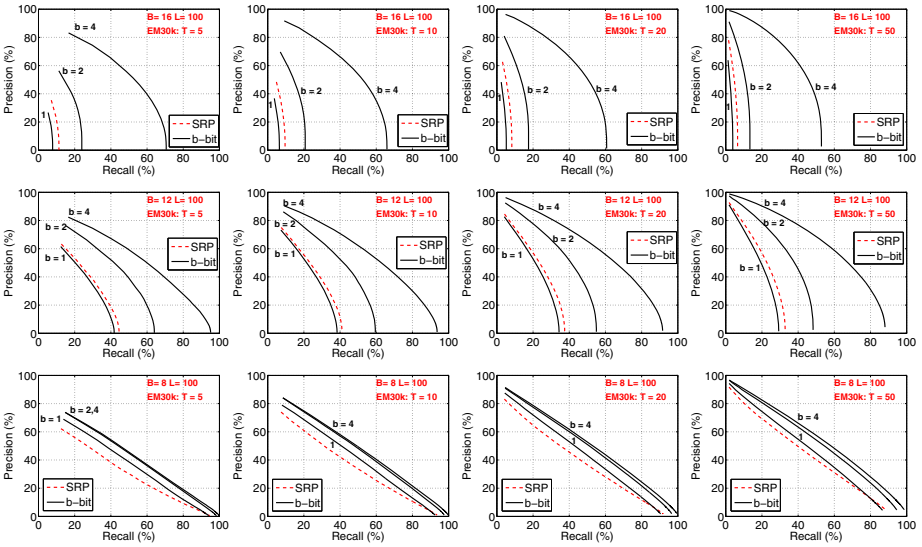


Fig. 14. Precision-Recall curves for SRP and b-bit minwise hashing on the *EM30k* dataset, using $L = 100$ tables

Figure 14 presents the precision-recall curves for $L = 100$ tables, again demonstrating the advantage of b-bit hashing over SRP.

5 Conclusion

This paper reports the first study of directly using the bits generated by b-bit minwise hashing to construct hash tables, for achieving sub-linear time near neighbor search in high-dimensional binary data. Our proposed scheme is extremely simple and exhibits superb performance compared to two strong baselines: spectral hashing (SH) and sign random projections (SRP).

Acknowledgement. This work is supported by NSF (DMS-0808864, SES-1131848), ONR (YIP-N000140910911), and DARPA (FA-8650-11-1-7149).

References

1. Tong, S.: Lessons learned developing a practical large scale machine learning system (2008), http://googleresearch.blogspot.com/2010/04/lessons-learned-developing-practical.html
2. Li, P., König, A.C.: b-bit minwise hashing. In: WWW, Raleigh, NC, 671–680 (2010)
3. Li, P., Shrivastava, A., Moore, J., König, A.C.: Hashing algorithms for large-scale learning. In: NIPS, Vancouver, BC (2011)
4. Broder, A.Z.: On the resemblance and containment of documents. In: The Compression and Complexity of Sequences, Positano, Italy, pp. 21–29 (1997)
5. Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the web. In: WWW, Santa Clara, CA, pp. 1157–1166 (1997)
6. Fetterly, D., Manasse, M., Najork, M., Wiener, J.L.: A large-scale study of the evolution of web pages. In: WWW, Budapest, Hungary, pp. 669–678 (2003)
7. Manku, G.S., Jain, A., Sarma, A.D.: Detecting Near-Duplicates for Web-Crawling. In: WWW, Banff, Alberta, Canada (2007)
8. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS (2008)
9. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of ACM 42(6), 1115–1145 (1995)
10. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: STOC, Montreal, Quebec, Canada, pp. 380–388 (2002)
11. Li, P., Hastie, T.J., Church, K.W.: Improving Random Projections Using Marginal Information. In: Lugosi, G., Simon, H.U. (eds.) COLT 2006. LNCS (LNAI), vol. 4005, pp. 635–649. Springer, Heidelberg (2006)
12. Friedman, J.H., Baskett, F., Shustek, L.: An algorithm for finding nearest neighbors. IEEE Transactions on Computers 24, 1000–1006 (1975)
13. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: STOC, Dallas, TX, pp. 604–613 (1998)
14. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM 51, 117–122 (2008)
15. Rajaraman, A., Ullman, J.: Mining of Massive Datasets, http://i.stanford.edu/ullman/mmds.html
16. Salakhutdinov, R., Hinton, G.E.: Semantic hashing. Int. J. Approx. Reasoning 50(7), 969–978 (2009)
17. Li, Z., Ning, H., Cao, L., Zhang, T., Gong, Y., Huang, T.S.: Learning to search efficiently in high dimensions. In: NIPS (2011)
18. Li, P.: Image classification with hashing on locally and gloablly expanded features. Technical report

A Variance-Space Comparisons (*b*-Bit Hashing v.s. SRP)

From the collision probability (5) of sign random projections (SRP), we can estimate the angle $\theta = \cos^{-1}\left(\frac{a}{\sqrt{f_1 f_2}}\right)$, with variance

$$\mathrm{Var}\left(\hat{\theta}\right) = \frac{\pi^2}{k}\left(1 - \frac{\theta}{\pi}\right)\left(\frac{\theta}{\pi}\right) = \frac{\theta(\pi - \theta)}{k}.$$

We can then estimate the intersection $a = |S_1 \cap S_2|$ by

$$\hat{a}_S = \cos\hat{\theta}\sqrt{f_1 f_2}, \qquad Var\left(\hat{a}_S\right) = \frac{\theta(\pi-\theta)}{k} f_1 f_2 \sin^2(\theta)$$

and the resemblance by $\hat{R}_S = \frac{\hat{a}_S}{f_1+f_2-\hat{a}_S}$

$$Var\left(\hat{R}_S\right) = \frac{\theta(\pi-\theta)}{k} f_1 f_2 \sin^2(\theta)\left(\frac{f_1+f_2}{(f_1+f_2-a)^2}\right)^2 + O\left(\frac{1}{k^2}\right).$$

We already know the variance of the b-bit minwise hashing estimator (4), denoted by $Var\left(\hat{R}_b\right)$. To compare it with $Var\left(\hat{R}_S\right)$, we define

$$W_b = \frac{Var\left(\hat{R}_S\right)}{Var\left(\hat{R}_b\right) \times b} = \frac{\theta(\pi-\theta) f_1 f_2 \sin^2(\theta)\left(\frac{f_1+f_2}{(f_1+f_2-a)^2}\right)^2}{\frac{[C_{1,b}+(1-C_{2,b})R][1-C_{1,b}-(1-C_{2,b})R]}{[1-C_{2,b}]^2}} \tag{12}$$

where $C_{1,b}$, $C_{2,b}$ (functions of r_1, r_2, b) are defined in (2). $W_b > 1$ means b-bit minwise hashing is more accurate than SRP at the same storage; see Figure 15.

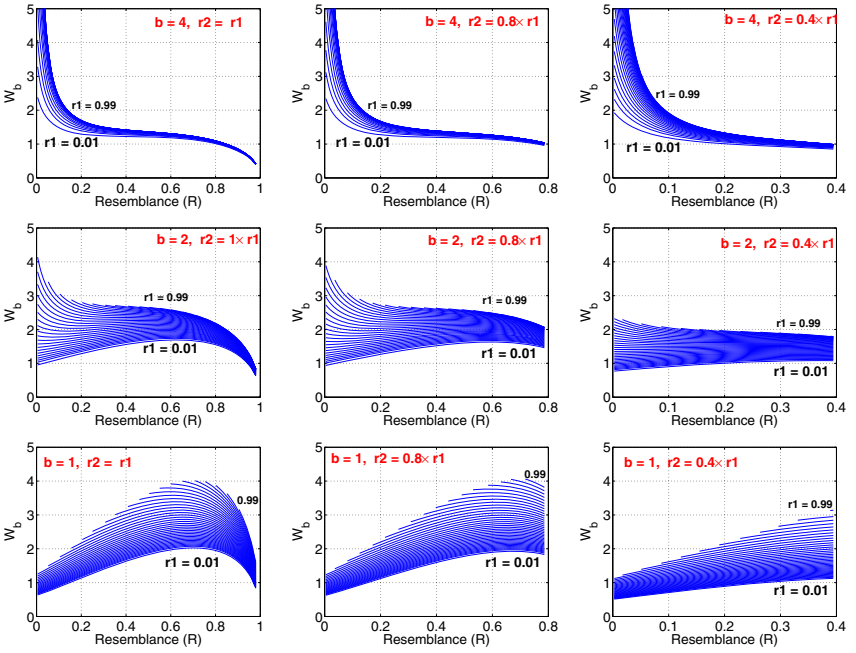


Fig. 15. W_b, $b = 4, 2, 1$, as defined in (12). r_1 and r_2 are defined in (2). Because $W_b > 1$ in most cases (sometimes significantly so), we know that b-bit minwise hashing is more accurate than 1-bit random projections at the same storage cost.

B The Derivation of (10)

$$\int_0^1 P_{b,k,L}(tR)dt = \int_0^1 1-\left(1-P_b^k(tR)\right)^L dt = 1-\int_0^1 \left(1-P_b^k(tR)\right)^L dt$$

$$=1-\int_0^1 \sum_{i=0}^{L}\binom{L}{i}(-1)^i P_b^{ki}(tR)dt = 1-\sum_{i=0}^{L}\binom{L}{i}(-1)^i \int_0^1 P_b^{ki}(tR)dt$$

$$=1-\sum_{i=0}^{L}\binom{L}{i}(-1)^i \frac{1}{2^{bki}}\int_0^1 \left(1+(2^b-1)tR\right)^{ki} dt$$

$$=1-\sum_{i=0}^{L}\binom{L}{i}(-1)^i \frac{1}{2^{bki}}\sum_{j=0}^{ki}(2^b-1)^j R^j \binom{ki}{j}\int_0^1 t^j dt$$

$$=1-\sum_{i=0}^{L}\binom{L}{i}(-1)^i \frac{1}{2^{bki}}\sum_{j=0}^{ki}\binom{ki}{j}(2^b-1)^j R^j \frac{1}{j+1}$$

$$=1-\sum_{i=0}^{L}\binom{L}{i}(-1)^i \frac{1}{2^{bki}}\frac{1}{(2^b-1)R}\frac{\left((2^b-1)R+1\right)^{ki+1}-1}{ki+1}$$

Fully Sparse Topic Models

Khoat Than[1] and Tu Bao Ho[1,2]

[1] Japan Advanced Institute of Science and Technology,
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
[2] John von Neumann Institute, Vietnam National University, HCM, Vietnam
{khoat,bao}@jaist.ac.jp

Abstract. In this paper, we propose Fully Sparse Topic Model (FSTM) for modeling large collections of documents. Three key properties of the model are: (1) the inference algorithm converges in linear time, (2) learning of topics is simply a multiplication of two sparse matrices, (3) it provides a principled way to directly trade off sparsity of solutions against inference quality and running time. These properties enable us to speedily learn sparse topics and to infer sparse latent representations of documents, and help significantly save memory for storage. We show that inference in FSTM is actually MAP inference with an implicit prior. Extensive experiments show that FSTM can perform substantially better than various existing topic models by different performance measures. Finally, our parallel implementation can handily learn thousands of topics from large corpora with millions of terms.

1 Introduction

Topic modeling has been increasingly maturing to be an attractive research area. Originally motivated from textual applications, it has been going beyond far from text to touch upon many amazing applications in Computer Vision, Bioinformatics, Software Engineering, Forensics, to name a few. Recently, much interest in this community has focused on developing topic models for large-scale settings, e.g., [1, 2, 3, 4, 5]. In our observations, the most common large-scale settings are: (a) *the number of training documents is large;* (b) *the number of topics to be learned is large;* (c) *the vocabulary size is large;* (d) *a large number of documents need to be a posteriori inferred in a limited time budget.* Further, combinations of these settings yield more challenges to the topic modeling community.

Most previous works have focused on the settings (a) and (b) by utilizing parallel/distributed/online architectures [1, 2, 3, 4, 6]. Those works, despite being breakthrough developments for Latent Dirichlet Allocation (LDA) [7], however will encounter some severe problems when vocabularies are very large or when new representations of documents have to be stored for doing other tasks. The main reason is that the Dirichlet distribution employed by LDA prevents any zero contributions of terms to topics and of topics to documents; therefore the learned topics and new representations of documents are extremely dense, consuming huge memory. This challenges deployment of LDA in practical applications with

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 490–505, 2012.

the settings (b) and (c).[1] Besides, inference methods for LDA are often slow, partially due to the NP-hardness nature of the model [8]. This characteristic may ruin out applicability of LDA to applications with the setting (d).

To reduce memory for efficient storage and inference, some studies have introduced the notion of sparsity for topic models. A popular approach is to use regularization techniques to impose sparsity constraints on topics or/and latent representations of documents, leading to RLSI [5], SRS [9], and STC [10].[2] Even though these models provide elegant solutions to the sparsity problem, they remain some serious drawbacks when dealing with large-scale settings. Indeed, SRS has no guarantee on convergence of inference/learning, and its scalability is unknown. STC is extremely problematic with learning, because learning of topics is to solve a optimization problem with a large number of variables which are inseparable. RLSI has high complexity for both learning and inference, triple in the number of topics. Finally, there are two common limitations of those models: first, auxiliary parameters of those models associated with regularization terms require us to do model selection, which is problematic in dealing with large-scale settings; second, one cannot directly trade off sparsity of solutions against time and quality.[3]

In this paper, we present our initial step towards resolving the mentioned four large-scale settings. Specifically, we present *Fully Sparse Topic Model* (FSTM) which is a simplified variant of LDA and Probabilistic Latent Semantic Analysis (PLSA) [13]. Unlike LDA, our model does not employ Dirichlet priors, and allows us to learn sparse latent representations of documents which are necessary for many applications, e.g., information/image retrieval. Inference in our model is casted as a concave maximization problem over the simplex of topics, which can be solved in linear time by the Frank-Wolfe algorithm [14].[4] One crucial property of the Frank-Wolfe algorithm is that it is easy to directly trade off sparsity level of solutions against quality and running time. So FSTM inherits this property for inference. Sparse inference in FSTM results in an interesting characteristic that there is an implicit prior over latent representations, without an explicit endowment even. In addition, learning of topics is formulated as an optimization problem so that it admits a closed-form solution, being a product of two sparse matrices. Hence the learned topics are very likely to be sparse.

Summarizing, the ability to learn sparse topics and to infer sparse latent representations of documents allows FSTM to save substantially memory for storage.

[1] Topical exploration of huge corpora, e.g. Google n-gram books, is an example.

[2] Another direction is to use Indian buffet processes [11] or a spike-and-slap distribution [12] to induce sparsity. Nonetheless, this approach often results in much involved models and thus complicates learning and inference. The proposed learning and inference methods [11, 12] are very far from a touch upon large-scale settings.

[3] For regularization techniques, one may expect to get sparser solutions by increasing the values of the auxiliary parameters. However, it is not always provably true. Hence such a control over sparsity is indirect.

[4] Note that our reformulation of inference for FSTM can be readily applied to many variants of PLSA and LDA, and hence can help accelerate their inference. The reason is that such models often assume a document to be a mixture of topics.

Table 1. Theoretical comparison of some topic models. V is the vocabulary size, K is the number of topics, n is the length of the document to be inferred. $\bar{K}$ is the average number of topics to which a term has nonzero contributions, $\bar{K} \leq K$. L is the number of iterations for inference. $\bar{K}$ (and L) is different for these models. '-' denotes *'no'* or *'unspecified'*; '✓' means *'yes'* or *'taken in consideration'*.

Model	FSTM	PLSA	LDA	STC	SRS	RLSI
Document sparsity	✓	-	-	✓	✓	-
Topic sparsity	✓	-	-	-	✓	✓
Sparsity control	direct	-	-	indirect	indirect	indirect
Trade-off:						
sparsity vs. quality	✓	-	-	-	-	-
sparsity vs. time	✓	-	-	-	-	-
Inference complexity	$L.O(n.\bar{K} + K)$	$L.O(n.K)$	$L.O(n.K)$	$L.O(n.K)$	$L.O(n.K)$	$L.O(V.\bar{K}^2 + K^3)$
Inference error	$O(1/L)$	-	-	-	-	0
Storage for topics	$V.\bar{K}$	$V.K$	$V.K$	$V.K$	$V.\bar{K}$	$V.\bar{K}$
Auxiliary parameters	0	0	0	3	2	2

Combined with a linear time inference algorithm, FSTM overcomes severe limitations of previous probabilistic models and can deal well with the settings (b), (c), and (d). Fast learning of topics and fast inference of documents also help FSTM to overcome limitations of non-probabilistic models (e.g., STC and RLSI), and enable us to deal well with the setting (a). Besides, an intriguing property of our model is the ability to directly trade off inference quality against running time and sparsity level of latent representations. This property is essential in order to resolve large-scale settings.

For further comparison, we report some theoretical characteristics of six closely related models in Table 1. Extensive experiments demonstrate that FSTM performs substantially better than various existing topic models by different performance measures. Our parallel implementation can handily learn thousands of topics from large corpora with millions of terms, which is on the order of magnitudes larger than known experiments with state-of-the-art models.

ROADMAP OF THIS PAPER: we discuss briefly in Section 2 some necessary concepts and results for concave optimization over simplex. The main model will be presented in Section 3. Section 4 is devoted to analyzing some theoretical characteristics of FSTM, and to revealing why there is an implicit prior over latent representations. Evaluation and comparison are discussed in details in Section 5. Our conclusions are in the final section.

2 Background

Before going deeply into our model and analysis, it is necessary to introduce some notations and to revisit some known results about sparse approximation for concave optimization over simplex.

$\mathcal{V}$: vocabulary of V terms, often written as $\{1, 2, ..., V\}$.

I_d: set of term indices of document $\boldsymbol{d}$,
i.e., each element in I_d is the vocabulary index of a term appearing in $\boldsymbol{d}$.

$\boldsymbol{d}$: a document represented as a count vector, $\boldsymbol{d} = (d_j)_{j \in I_d}$,
where d_j is the frequency of term j in $\boldsymbol{d}$.

$\mathcal{C}$: a corpus consisting of M documents, $\mathcal{C} = \{\boldsymbol{d}_1, ..., \boldsymbol{d}_M\}$.
$\boldsymbol{\beta}_k$: a topic which is a distribution over the vocabulary $\mathcal{V}$.
$\boldsymbol{\beta}_k = (\beta_{k1}, ..., \beta_{kV})^t$, $\beta_{kj} \geq 0, \sum_{j=1}^{V} \beta_{kj} = 1$.
K: number of topics.

A topic model often assumes that a given corpus is composed from K topics, $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K)$, and each document is a mixture of those topics. Example models include PLSA, LDA and many of their variants. Under those models, each document has another latent representation. Such latent representations of documents can be inferred once those models have been learned previously.

Definition 1 (Topic proportion). *Consider a topic model $\mathfrak{M}$ with K topics. Each document $\boldsymbol{d}$ will be represented by $\boldsymbol{\theta} = (\theta_1, ..., \theta_K)^t$, where θ_k indicates the proportion that topic k contributes to $\boldsymbol{d}$, and $\theta_k \geq 0, \sum_{k=1}^{K} \theta_k = 1$. $\boldsymbol{\theta}$ is called* topic proportion *(or latent representation) of $\boldsymbol{d}$.*

Definition 2 (Inference). *Consider a topic model $\mathfrak{M}$ with K topics, and a given document $\boldsymbol{d}$. The inference problem is to find the topic proportion that maximizes the likelihood of $\boldsymbol{d}$ under the model $\mathfrak{M}$.*

For some applications, it is necessary to infer which topic contributes to a specific emission of a term in a document. Nevertheless, it may be unnecessary for many other applications. Therefore we do not take this problem into account and leave it for future work.

Definition 3 (Document sparsity). *Consider a topic model $\mathfrak{M}$ with K topics, and a corpus $\mathcal{C}$ with M documents. Let $\boldsymbol{\theta}_m$ be the topic proportion of document $\boldsymbol{d}_m \in \mathcal{C}$. Then the* document sparsity *of $\mathcal{C}$ under the model $\mathfrak{M}$ is defined as the proportion of non-zero entries of the new representation of $\mathcal{C}$, i.e., document sparsity $= \frac{\#\textit{non-zeros of } (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M)}{M.K}$.*

Definition 4 (Topic sparsity). *Consider a topic model $\mathfrak{M}$ with K topics $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K)$.* Topic sparsity *of $\mathfrak{M}$ is defined as the proportion of non-zero entries in $\boldsymbol{\beta}$, i.e., topic sparsity $= \frac{\#\textit{non-zeros of } \boldsymbol{\beta}}{V.K}$.*

2.1 Concave Maximization over Simplex and Sparse Approximation

Let $\boldsymbol{b}_1, ..., \boldsymbol{b}_K$ be vectors in $\mathbb{R}^V$. Denote as $\Delta = conv(\boldsymbol{b}_1, ..., \boldsymbol{b}_K)$ the convex hull of those vectors. Consider a concave function $f(\boldsymbol{x}) : \mathbb{R}^V \to \mathbb{R}$ which is twice differentiable over Δ. We are interested in the following problem, *concave maximization over simplex*,

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \Delta} f(\boldsymbol{x}) \tag{1}$$

Convex/concave optimization has been extensively studied in the optimization literature. There has been various excellent results such as [15, 16]. However, we will concentrate on sparse approximation algorithms specialized for the problem (1). More specifically, we focus on the Frank-Wolfe algorithm [14].

Loosely speaking, the Frank-Wolfe algorithm is an approximation one for the problem (1). Starting from a vertex of the simplex Δ, it iteratively selects the most potential vertex of Δ to change the current solution closer to that vertex in order to maximize $f(\boldsymbol{x})$. It has been shown that the Frank-Wolfe algorithm converges at a linear rate to the optimal solution. Moreover, at each iteration, the algorithm finds a provably good approximate solution lying in a face of Δ.

Theorem 1. *[14] Let f be a twice differentiable concave function over Δ, and denote $C_f = -\frac{1}{2}\sup_{\boldsymbol{y},\boldsymbol{z}\in\Delta;\tilde{\boldsymbol{y}}\in[\boldsymbol{y},\boldsymbol{z}]}(\boldsymbol{y}-\boldsymbol{z})^t.\nabla^2 f(\tilde{\boldsymbol{y}}).(\boldsymbol{y}-\boldsymbol{z})$. After ℓ iterations, the Frank-Wolfe algorithm finds a point $\boldsymbol{x}_\ell$ on an $(\ell+1)-$dimensional face of Δ such that*

$$\max_{\boldsymbol{x}\in\Delta} f(\boldsymbol{x}) - f(\boldsymbol{x}_\ell) \leq \frac{4C_f}{\ell+3}. \tag{2}$$

It is worth noting some observations about the Frank-Wolfe algorithm:

- It achieves a linear rate of convergence, and has provably bounds on the goodness of solutions. These are crucial for practical applications.
- Overall running time mostly depends on how complicated f and ∇f are.
- It provides an explicit bound on the dimensionality of the face of Δ in which an approximate solution lies. After ℓ iterations, $\boldsymbol{x}_\ell$ is a convex combination of at most $\ell+1$ vertices of Δ. Let $\boldsymbol{\theta}_\ell$ be the coefficients of that combination, i.e., $\boldsymbol{x}_\ell = \sum_k \theta_{\ell k}\boldsymbol{b}_k$. Theorem 1 ensures that at most $\ell+1$ out of K components of $\boldsymbol{\theta}_\ell$ are non-zero. This implies that we can find an approximate solution to the problem (1) with an associated sparse *latent representation* $\boldsymbol{\theta}_\ell$.
- It is easy to directly control the sparsity level of such latent representations by trading off sparsity against quality. (The fewer the number of iterations, the more sparse the latent representation.) This characteristic makes the algorithm more attractive for resolving high dimensional problems.

3 Fully Sparse Topic Models

In this section, we present our model, named *Fully Sparse Topic Model* (FSTM), which is considerably simple. To be more detailed, FSTM assumes that a corpus is composed from K topics, $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K$, and each document $\boldsymbol{d}$ is generated by the following process:

1. *Pick randomly a topic proportion $\boldsymbol{\theta}$.*
2. *For the jth word in $\boldsymbol{d}$:*
   - *Pick a latent topic z_k with probability $P(z_k|\boldsymbol{d}) = \theta_k$,*
   - *Generate a word w_j with probability $P(w_j|z_k) = \beta_{kj}$.*

It is straightforward to see that FSTM is a simplified variant of LDA. The main difference is that FSTM does not employ Dirichet prior over topic proportions, and deliberately allows only few topics to contribute to a document. This relaxation allows us to infer really sparse topic proportions of documents. Besides, we further propose an approach to learning topics so that sparsity of topic proportions can be exploited. The latent topics are sparse as well, hence leading to

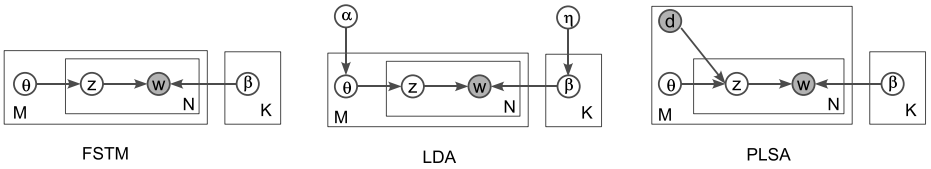


Fig. 1. Graphical representations of three topic models

the name of our model. Figure 1 depicts the graphical representation of FSTM, accompanied by PLSA and LDA.

In spite of no explicit prior over $\boldsymbol{\theta}$ in the model description, we will see in Section 4 that in fact there exists an implicit prior having density function $p(\boldsymbol{\theta}|\lambda) \propto \exp(-\lambda.||\boldsymbol{\theta}||_0)$, where $||\boldsymbol{\theta}||_0$ is the number of non-zero entries of $\boldsymbol{\theta}$. This property is a consequence of sparse inference in our model. Note that this property of FSTM is intriguing and hence we term it *"implicit modeling"*.

3.1 Inference

Given a document $\boldsymbol{d}$ and topics $\boldsymbol{\beta}$, the inference task in FSTM is to find which topics contribute to $\boldsymbol{d}$ and how much they contribute to $\boldsymbol{d}$. In other words, we have to infer $\boldsymbol{\theta}$. Unlike existing inference approaches for topic models, we will not make effort to infer directly $\boldsymbol{\theta}$. Instead, we reformulate the inference task as a concave maximization problem over the simplex of topics.

Lemma 1. *Consider FSTM with topics* $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K$*, and a given document* $\boldsymbol{d}$*. The inference problem can be reformulated as the following concave maximization problem, over the simplex* $\Delta = conv(\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K)$*,*

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \Delta} \sum_{j \in I_d} d_j \log x_j. \tag{3}$$

Proof. For a given document $\boldsymbol{d}$, the probability that a term w_j appears in $\boldsymbol{d}$ can be expressed as $P(w_j|\boldsymbol{d}) = \sum_{k=1}^{K} P(w_j|z_k).P(z_k|\boldsymbol{d}) = \sum_{k=1}^{K} \theta_k \beta_{kj}$. Hence the log likelihood of $\boldsymbol{d}$ is $\log P(\boldsymbol{d}) = \log \prod_{j \in I_d} P(w_j|\boldsymbol{d})^{d_j} = \sum_{j \in I_d} d_j \log P(w_j|\boldsymbol{d}) = \sum_{j \in I_d} d_j \log \sum_{k=1}^{K} \theta_k \beta_{kj}$.

The inference task is the problem of searching for $\boldsymbol{\theta}$ to maximize the likelihood of $\boldsymbol{d}$. Denoting as $x_j = \sum_{k=1}^{K} \theta_k \beta_{kj}$ and $\boldsymbol{x} = (x_1, ..., x_V)^t$, we arrive at

$$\log P(\boldsymbol{d}) = \sum_{j \in I_d} d_j \log x_j. \tag{4}$$

Therefore optimization over $\boldsymbol{\theta}$ now is translated into that over $\boldsymbol{x}$. Note that $\boldsymbol{x} = (x_1, ..., x_V)^t = \sum_{k=1}^{K} \theta_k \boldsymbol{\beta}_k$. Combining this with the fact that $\sum_k \theta_k = 1$, $\theta_k \geq 0, \forall k$, one can easily realize that $\boldsymbol{x}$ is a convex combination of the K topics $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K$. It implies $\boldsymbol{x} \in \Delta$. As a result, the inference task is in turn the problem of finding $\boldsymbol{x} \in \Delta$ that maximizes the objective function (4). □

Algorithm 1. Inference algorithm

Input: document $\boldsymbol{d}$ and topics $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K$.
Output: $\boldsymbol{\theta}_*$, for which $\sum_{k=1}^{K} \theta_{*,k}\boldsymbol{\beta}_k = \boldsymbol{x}_*$ maximizes $f(\boldsymbol{x}) = \sum_{j \in I_d} d_j \log x_j$.
Pick as $\boldsymbol{\beta}_r$ the vertex of $\Delta = conv(\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K)$ with largest f value.
Set $\boldsymbol{x}_0 := \boldsymbol{\beta}_r; \theta_{0,r} = 1; \theta_{0,k} = 0, \forall k \neq r$;
for $\ell = 0, ..., \infty$ **do**
  $i' := \arg\max_i \boldsymbol{\beta}_i^t \nabla f(\boldsymbol{x}_\ell)$;
  $\alpha' := \arg\max_{\alpha \in [0,1]} f(\alpha\boldsymbol{\beta}_{i'} + (1-\alpha)\boldsymbol{x}_\ell)$;
  $\boldsymbol{x}_{\ell+1} := \alpha'\boldsymbol{\beta}_{i'} + (1-\alpha')\boldsymbol{x}_\ell$;
  $\boldsymbol{\theta}_{\ell+1} := (1-\alpha')\boldsymbol{\theta}_\ell$; and then set $\theta_{\ell+1,i'} := \theta_{\ell+1,i'} + \alpha'$.
end for

This lemma provides us a connection between inference and concave optimization, and allows us to seamlessly use the Frank-Wolfe algorithm for inference. An appropriate adaptation to the Frank-Wolfe algorithm [14] results in an inference algorithm for FSTM, as presented in Algorithm 1. In our implementation, we solve for α by the gradient ascent approach.

3.2 Learning

The task of learning FSTM is to learn all topics $\boldsymbol{\beta}$, given a corpus $\mathcal{C}$. We use EM scheme to iteratively learn the model. Specifically, we repeat the following two steps until convergence: *(E-step)* do inference for each document of $\mathcal{C}$; *(M-step)* maximize the likelihood of $\mathcal{C}$ with respect to $\boldsymbol{\beta}$.

Note that the E-step for each document is discussed in the previous subsection. The remaining task is to solve for $\boldsymbol{\beta}$. Denoting as $\boldsymbol{\theta}_d$ the topic proportion of document $\boldsymbol{d} \in \mathcal{C}$ which has been inferred in the E-step, we express the log likelihood of $\mathcal{C}$ as $\log P(\mathcal{C}) = \sum_{\boldsymbol{d} \in \mathcal{C}} \log P(\boldsymbol{d}) = \sum_{\boldsymbol{d} \in \mathcal{C}} \sum_{j \in I_d} d_j \log \sum_{k=1}^{K} \theta_{dk}\beta_{kj} \geq \sum_{\boldsymbol{d} \in \mathcal{C}} \sum_{j \in I_d} d_j \sum_{k=1}^{K} \theta_{dk} \log \beta_{kj}$. We have used Jensen's inequality to derive the last term, owing to the fact $\sum_k \theta_{dk} = 1, \theta_{dk} \geq 0, \forall k$. Next we maximize the lower bound of $\log P(\mathcal{C})$ with respect to $\boldsymbol{\beta}$. In other words, we have to maximize

$$g(\boldsymbol{\beta}) = \sum_{\boldsymbol{d} \in \mathcal{C}} \sum_{j \in I_d} d_j \sum_{k=1}^{K} \theta_{dk} \log \beta_{kj}, \text{ such that } \sum_{j=1}^{V} \beta_{kj} = 1, \beta_{kj} \geq 0, \forall k, j. \quad (5)$$

It is worthwhile noticing that the vectors $\boldsymbol{\beta}_k$ are separable from each other in the objective function $g(\boldsymbol{\beta})$. Hence we can solve for each individually. Taking the Lagrange function into consideration and forcing its derivatives to be 0, we easily arrive at the following solution

$$\beta_{kj} \propto \sum_{\boldsymbol{d} \in \mathcal{C}} d_j \theta_{dk}. \quad (6)$$

Up to this point, we can learn FSTM by iterating E-step and M-step until convergence. In the E-step, each document is inferred by using the Frank-Wolfe algorithm, given the objective function as in (3) and topics $\boldsymbol{\beta}$. The M-step only does simple calculation according to (6) to update all topics.

4 Theoretical Analysis

We will show that the inference algorithm for FSTM can provide provably good solutions. It requires modestly few arithmetic operations, linear in the length of the document to be inferred or/and in the number of topics. Further, we can easily trade off quality of solution against sparsity and inference time. Existing topic models do not own these interesting properties.

4.1 Complexity and Goodness of Inference

Theorem 2. *Consider FSTM with K topics, and a document $\boldsymbol{d}$. Let C_f be defined as in Theorem 1 for the function $f(\boldsymbol{x}) = \sum_{j \in I_d} d_j \log x_j$. Then Algorithm 1 converges to the optimal solution with a linear rate. In addition, after L iterations, the inference error is at most $4C_f/(L+3)$, and the topic proportion $\boldsymbol{\theta}$ has at most $L+1$ non-zero components.*

Proof. Inference of FSTM is exactly the Frank-Wolfe algorithm for the function $f(\boldsymbol{x}) = \sum_{j \in I_d} d_j \log x_j$ which is twice differentiable at all $\boldsymbol{x}$ satisfying $x_j > 0$, $\forall j \in I_d$. Hence this theorem is a corollary of Theorem 1. □

Next we will analyze computational complexity of the inference algorithm. Common technique to store a sparse matrix is row-wise, i.e., we store all non-zero elements in a row of that matrix by an 1-dimensional array. This is beneficial to do multiplication of a sparse matrix with a vector. Indeed, consider a matrix $\boldsymbol{B}$ of size $m \times n$. Letting $\bar{m}$ be the average number of non-zero elements of a column of $\boldsymbol{B}$, computing $\boldsymbol{Bx}$ requires only $O(n.\bar{m} + m)$ arithmetic operations.

Theorem 3. *Each iteration of Algorithm 1 requires only $O(n.\bar{K}+K)$ arithmetic operations, where $\bar{K}$ is the average number of topics to which a term has non-zero contributions, $\bar{K} \leq K$, and $n = |I_d|$. Overall, after L iterations, Algorithm 1 requires $L.O(n.\bar{K} + K)$ arithmetic operations.*

Proof. Letting $\boldsymbol{a} = \nabla f(\boldsymbol{x})$, we have $\boldsymbol{\beta}^t \nabla f(\boldsymbol{x}) = \boldsymbol{\beta}^t \boldsymbol{a}$. Note that $\boldsymbol{a}$ is very sparse because of $a_i = \partial f / \partial x_i = 0$, for $i \notin I_d$. Hence only n columns of $\boldsymbol{\beta}^t$ involve in computation of $\boldsymbol{\beta}^t \boldsymbol{a}$. This implies that we need just $O(n.\bar{K} + K)$ arithmetic operations to compute $\boldsymbol{\beta}^t \boldsymbol{a}$ and to find the index i'. $O(n.\bar{K} + K)$ arithmetic operations are also sufficient to do the initial step of choosing $\boldsymbol{x}_0$, since the most expensive computations are to evaluate f at the vertices of the simplex, which amounts to a multiplication of $(\log \boldsymbol{\beta})^t \boldsymbol{d}$, where $\log \boldsymbol{\beta} = (\log \beta_{ij})_{V \times K}$.

Searching for α can be done very quickly since the problem is concave in one variable. Each evaluation of $f(\boldsymbol{x})$ requires only $O(n)$ operations. Moreover $O(n.\bar{K} + K)$ arithmetic operations are sufficient to update other variables. □

Remark 1 (Learning). Our model is learned by the EM scheme. Each EM iteration requires $M.L.O(n.\bar{K} + K)$ arithmetic operations to infer M training documents, and an update for the topics according to formula (6). Note that update of topics amounts to multiplication of two very sparse matrices (one is the matrix representing the training corpus, and the other is the new representation of that corpus.) Hence it can be computed very fast.

4.2 Managing Sparsity Level and Trade-off

Good solutions are often necessary for practical applications. In practice, we may have to spend intensive time and huge memory to search such solutions. This sometimes is not necessary or impossible in limited time/memory settings. Hence one would prefer to trading off quality of solutions against time/memory.

Searching for sparse solutions is a common approach in Machine Learning to reduce memory for storage and efficient processing. Most previous works have tried to learn sparse solutions by imposing regularization which induces sparsity, e.g., L1 regularization [10, 5] and entropic regularization [9]. Nevertheless, those techniques are severely limited in the sense that we cannot directly control sparsity level of solutions (e.g., one cannot decide how many non-zero components solutions should have). In other words, sparsity level of solutions is a priori unpredictable. This limitation makes regularization techniques inferior in memory limited settings. This is also the case with other works that employ some probabilistic distributions to induce sparsity such as [11, 12].

Unlike prior topic models, the inference algorithm for FSTM naturally provides a principled way to control sparsity. Theorem 2 implies that if stopped at the Lth iteration, the inferred solution has at most $L+1$ non-zero components. Hence one can control sparsity level of solutions by simply limiting the number of iterations. It means that we can predict a priori how sparse and how good the inferred solutions are. Less iterations, more sparse (but probably worse) solutions of inference. Besides, we can trade off sparsity against inference time. More iterations imply more necessary time and probably denser solutions.

4.3 Implicit Prior over $\boldsymbol{\theta}$

In Section 3 we describe our model without any specific prior over latent representations $\boldsymbol{\theta}$. As well-known in the literature, no prior endowment may cause a model to be prone to overfitting. Nonetheless, it seems not the case with FSTM. Indeed, we argue that there is an implicit prior over $\boldsymbol{\theta}$ in the model.

Note that the inference algorithm of FSTM allows us to easily trade off sparsity of solutions against quality and time. If one insists on solutions with at most t nonzero components, the inference algorithm can modified accordingly. In this case, it mimics that one is trying to find a solution to the problem $\max_{\boldsymbol{\theta}\in\Delta_1}\{f(\boldsymbol{\theta}) : ||\boldsymbol{\theta}||_0 \leq t\}$, where Δ_1 is the unit simplex in $\mathbb{R}^K$. We remark a well-known fact that the constraint $||\boldsymbol{\theta}||_0 \leq t$ is equivalent to addition of a penalty term $\lambda.||\boldsymbol{\theta}||_0$ to the objective function [17], for some constant λ. Therefore, one is trying to solve for $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\Delta_1}\{f(\boldsymbol{\theta}) - \lambda.||\boldsymbol{\theta}||_0\} = \arg\max_{\boldsymbol{\theta}\in\Delta_1} P(\boldsymbol{d}|\boldsymbol{\theta}).P(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}\in\Delta_1} P(\boldsymbol{\theta}|\boldsymbol{d})$, where $p(\boldsymbol{\theta}) \propto \exp(-\lambda.||\boldsymbol{\theta}||_0)$. Notice that the last problem, $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\Delta_1} P(\boldsymbol{\theta}|\boldsymbol{d})$, is an MAP inference problem. Hence, these observations basically show that inference by Algorithm 1 for sparse solutions mimics MAP inference. As a result, there exists an implicit prior, having density function $p(\boldsymbol{\theta};\lambda) \propto \exp(-\lambda.||\boldsymbol{\theta}||_0)$, over latent topic proportions. This is another characteristic that distinguish FSTM from existing topic models.

5 Experimental Evaluation

This section is devoted to investigating practical performance of our model. Due to space limit, we focus mainly on investigating practical behaviors of FSTM to see clearly its characteristics. We will describe briefly performance of our model on huge corpora, omitting implementation details in this extended abstract.[5]

5.1 Sparsity, Time, Quality, and Trade-off

We first aim at answering the following questions: (1) *how sparse are topics and latent representations of documents?* (2) *how fast can the model infer/learn?* (3) *can the model achieve good quality?* To this end, we chose 4 corpora for experiments: 2 small (AP, KOS), and 2 average (Grolier, Enron).[6] Figure 2 contains some information about these corpora. For each corpus, we used 90% for learning and 10% held out for evaluation. Four models are included for comparison: FSTM, PLSA, LDA, and STC.[7] In our experiments we used the same convergence criteria for these models: relative improvement of log likelihood (or objective functions in STC) is less than 10^{-6} for inference, and 10^{-4} for learning; at most 1000 iterations are allowed to do inference. We used default settings for some other auxiliary parameters of STC, relating to regularization terms.

Document sparsity: Figure 2 presents the results of experiments on four corpora. Document sparsity is used to see sparsity level of latent representations discovered by those models. Observing the first two rows of Figure 2, one can see that all models, except LDA, can discover sparse latent representations. PLSA interestingly can discover very sparse representations. It even often outperformed STC, which was intentionally designed for modeling sparsity. However, it seems that PLSA achieved sparse solutions by incident. Indeed, we rarely observed sparse topic proportions in the learning phase, but inference often resulted in sparse ones. One crucial reason for these contrary behaviors is that information was lost when saving the learned models, as we observed many nonzero elements of topics went to 0. STC can indeed discover sparse latent representations as expected. Nonetheless, the discovered sparsity level was not very high, i.e., new representations of documents were still pretty dense. Furthermore, the sparsity level seems to be inconsistent as the number of topics increases

On contrary, FSTM can discover very sparse latent representations in both learning and inference phases. The sparsity level consistently decreases as the

[5] The code is available at http://www.jaist.ac.jp/~s1060203/codes/fstm.

[6] AP was retrieved from http://www.cs.princeton.edu/~blei/lda-c/ap.tgz
KOS and Enron were retrieved from http://archive.ics.uci.edu/ml/datasets/
Grolier was from http://cs.nyu.edu/~roweis/data.html

[7] LDA code was taken from http://www.cs.princeton.edu/~blei/lda-c/
STC code was taken from http://www.cs.cmu.edu/~junzhu/stc/
PLSA was coded by ourselves with the best effort. SRS and RLSI were not included because of two reasons. First, there is no available code for these models. More importantly, there is an inconsistence in the update formula derived in [9] that prevents us from implementation; RLSI heavily needs involved distributed architectures.

Data	AP	KOS	Enron	Grolier
M	2246	3430	39861	29762
V	10473	6906	28102	15276

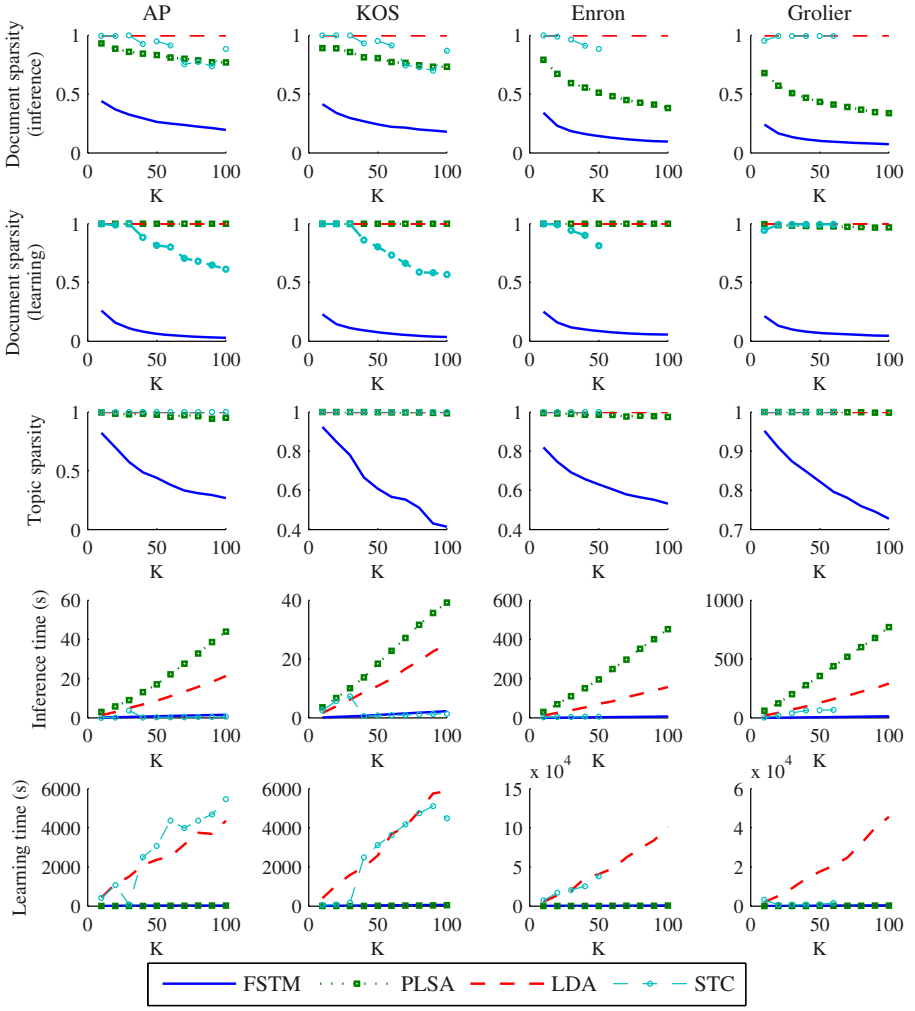


Fig. 2. Experimental results as the number K of topics increases. For STC, there was a memory problem when dealing with Enron and Grolier for large K (e.g., when $K = 70$, STC has to solve a optimization problem with more than 20 millions of variables, and hence cannot be handled in a personal PC.) Hence we could not do experiments for such large K's.

number of topics increases. This implies that despite modeling a corpus with many topics, few topics actually contributes to a specific document. For example, on average, only 3 topics have non-zero contributions to a document of AP among 100 topics of the model; when modeling with 10 topics, only 2 topics on average have non-zero contributions to a document. This seems to be consistent with the fact that a document often says about few topics, independent with the number of topics a model is taking into account. Hence FSTM can discover very compact representations and save significantly memory for storage.

Topic sparsity: observing Figure 2, one easily realizes that most models could not discover sparse topics. LDA and STC are not surprised, because topics are assumed to be samples of Dirichlet distributions which implicitly prevent any zero contribution of terms to topics. PLSA could discover some sparse topics, but the sparsity level was insignificant. FSTM outperformed other models in this aspect, having discovered very sparse topics. The sparsity level of topics tends to increase as we model data with more topics. This achievement can be explained by the facts that new representations of documents inferred by FSTM are very sparse, that the original documents are sparse, and that topics are simply a product of these two sparse representations (see equation 6). Therefore, the learned models are often significantly compact.

Inference time: in Section 4, we have shown theoretically that inference of FSTM is in linear time. This is further supported by our experiments, as depicted in Figure 2. Both FSTM and STC worked comparably in practice. PLSA inferred most slowly by the folding-in technique. LDA can infer much more quickly by fast variational Bayesian methods [7]. Nevertheless, it still worked much more slowly than FSTM, often tens of times more slowly. There are at least two reasons for this slow inference: first, the inference problem in LDA is inherently NP-hard [8] and thus may require much time to reach at good solutions; second, the variational Bayesian algorithm has to do many computations relating to logarithm, exponent, gamma, and digamma functions which are expensive. In contrast, inference in FSTM can be done in linear time, and the objective function (likelihood) is relatively cheap to compute. In addition, the learned topics are often very sparse. All of these contribute to speeding up inference in FSTM.

Learning time: observing the last row of Figure 2, one can see that LDA and STC learned really slowly, often hundreds/thousands of times more slowly than FSTM and PLSA.[8] Slow learning of STC can be explained by the fact that learning of topics in this model is very expensive, since we have to solve a optimization problem with huge number, $K.V$, of variables which are inseparable. LDA learned slowly because its inference algorithm is slow, and it has to solve various optimization problems requiring various evaluations of Gamma and Digamma functions which are often expensive. PLSA learned fastest due

[8] At some settings, we observe that STC did stop learning very early after only 4 or 5 iterations, but inference after that paid more time to do than usual. Otherwise, it needed many iterations (often more than 30) to reach convergence. Hence we suppose that those early terminations were caused by some internal issues.

to its simple learning formulations. There is a seemingly contrary behavior of PLSA, in which learning is fastest but inference is slowest. The main reason is that inference by folding-in [13] is an adaptation of learning, and more importantly learning does not require doing separately inference of documents which differs from other models. FSTM can learn very fast, comparably with PLSA. One reason for such a fast learning is the fast inference algorithm. Another reason is that the inferred topic proportions and topics themselves are very sparse, and hence help further speed up learning.

Quality: we next consider how good our model is. We use three measures to quantify the quality: Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC) [18], and Perplexity [7]. BIC and AIC are popular measures for model selection in Machine Learning.[9] They measure both simplicity and goodness-of-fit of the considered models; the simpler is preferred when two models have comparable quality of fitting data. A model with larger BIC/AIC is more likely to overfit the data [18]. Perplexity is also a common measure in topic modeling literature to compare predictive power of different models.[10]

Figure 3 presents the quality of three models on four corpora. (STC was not included in this investigation, because the objective function in learning is a regularized one, and hence different in manner with probabilistic topic models.) Observing the first two rows of the figure, one can easily realize that BIC and AIC of FSTM were significantly better than those of LDA and PLSA for most experiments. Note that FSTM can learn very sparse topics as previously discussed. In addition, we observed that the likelihoods achieved by FSTM were often comparable with those by PLSA, while those by LDA were often worst. Hence FSTM was evaluated better than other models according to BIC/AIC. For PLSA and LDA, despite using more free parameters (dense topics) to model data, the achieved likelihoods were not very significantly greater than those of FSTM. Therefore, they are more likely prone to overfitting. The ability to avoid overfitting of FSTM in these experiments supports further the theoretical analysis in Section 4, where an implicit prior is argued to keep FSTM from overfitting.

The last row of Figure 3 shows perplexity obtained by three models. We observe that PLSA consistently achieved better perplexity than LDA and FSTM. This seems unusual since LDA is a Bayesian extension of PLSA and thus should have better predictive power. Nonetheless, in our observations, at least two factors had contributed to this inferior predictiveness: first, the variational Bayesian method [7] is not guaranteed to find good solutions; second, the objective of inference in LDA is posterior probability $P(\boldsymbol{\theta}|\boldsymbol{d})$, not the likelihood $P(\boldsymbol{d})$, while perplexity is mainly about likelihood. FSTM achieved good predictive power.

[9] $AIC = (-2\log\mathfrak{L} + 2p)/M$, and $BIC = (-2\log\mathfrak{L} + p\log M)/M$, where $\mathfrak{L}$ is the achieved likelihood, and p is the number of free parameters of the model. Note that free parameters in the considered topic models basically correspond to the entries of topics, and one more for LDA. Hence $p = (V-1)K + 1$ for LDA, while $p + K$ for FSTM/PLSA is the number of non-zero entries of the learned topics.

[10] Perplexity of a model $\mathfrak{M}$ is calculated on the testing set $\mathcal{D}$ by $Perp(\mathcal{D}|\mathfrak{M}) = \exp\left(-\sum_{\boldsymbol{d}\in\mathcal{D}}\log P(\boldsymbol{d}|\mathfrak{M})/\sum_{\boldsymbol{d}\in\mathcal{D}}|\boldsymbol{d}|\right)$.

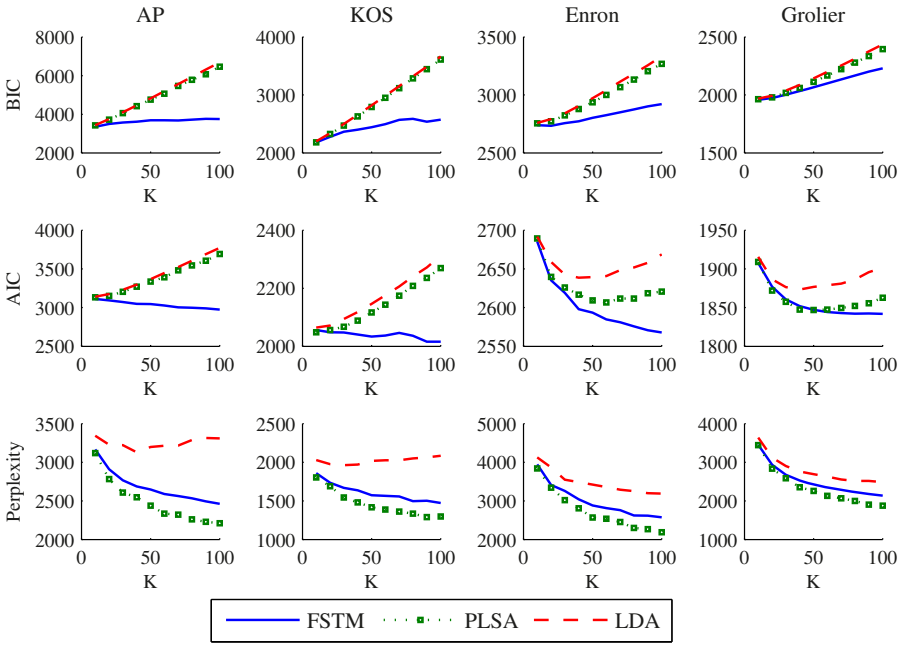


Fig. 3. Quality of three models as the number of topics increases. Lower is better.

The inference algorithm of FSTM played a crucial role in this good power, since it is guaranteed to find provably good solutions as analyzed in Section 4.

Trade-off: Figure 4 illustrates how FSTM trades off sparsity of solutions against inference quality (measured by perplexity) and running time. Unsurprisingly, more iterations means better quality but probably denser topic proportions. Note that the upper bound on inference error in Theorem 2 is quite loose. However, in practice inference converged very quickly, as observed in Figure 4. After 20 iterations on average, the quality and sparsity level were almost stable. We rarely observed inference needed more than 100 iterations to reach convergence. This is an interesting behavior of FSTM and is appealing to resolving large-scale settings.

5.2 Large-Scale Settings

We implemented a parallel version of FSTM using OpenMP for large-scale learning. Even though OpenMP is a shared memory model, we employed both data parallelism and task parallelism schemes. Data is distributed across clusters of CPUs, each cluster has its own subset of data and sub-model in the learning phase. Communication of a cluster with the master is only its sub-model. Note that FSTM consistently learns sparse models. Hence communication of sub-models for FSTM are significantly more compact than other implementations of LDA [1, 3, 4]. (Details of implementation are omitted due to space limit.)

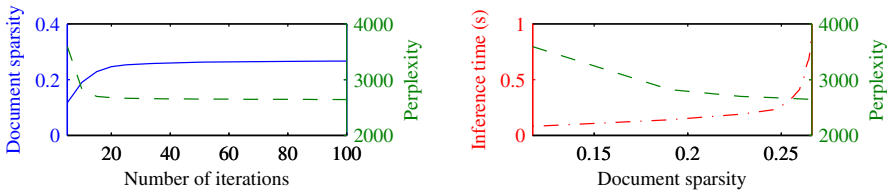


Fig. 4. Illustration of trading off sparsity against quality and time. Inference was done on AP, where FSTM had been learned with 50 topics.

We then experimented with the Webspam corpus consisting of 350K documents with more than 16 millions of terms.[11] 2000 topics was selected, and we run on 128 CPUs (each with 2.9 GHz), divided into 32 clusters. We observed that even though the documents in this corpus are often very long, inference was done very quickly, and each iteration of the EM algorithm took approximately 1 hour. After convergence, the achieved topic sparsity is 0.0114 and document sparsity is 0.0028. This means, over 2000 topics, on average only 5.6 topics contribute to a specific document; and 1.14% of 16 million terms significantly contribute to a topic. Storage of the new representation of the corpus is less than 34Mb, substantially reduced from 23.3Gb of the original one.

Since Webspam is a supervised dataset, we did a classification experiment either. We use the new representation of the corpus previously learned by FSTM to be the input for Liblinear [19], resulting in Liblinear+FSTM method for classification where FSTM plays the role as a dimensionality reduction subroutine. Using 5-folds cross-validation and default settings for Liblinear, the obtained accuracy is 99.146%. The most recent advanced method [20] can achieve a comparable accuracy of 99.15%, but evaluated on only one split of data. Note that the new representation has 2000 dimensions, and is 700 times smaller than the original one. All of these suggest that FSTM can infer very meaningful representations of documents. As a result, FSTM can provide us a useful tool, not only a model of linguistic data but also a dimensionality reduction approach, to efficiently deal with large-scale settings.

6 Conclusion

We have introduced our novel topic model for modeling large collections of documents. Our model overcomes many serious limitations of existing topic models, and has been demonstrated to work qualitatively on real data. The scalability of our model enables us to easily deal with large-scale settings.

Our work in this paper also touches upon two interesting questions: (1) *Is there an algorithm for efficiently inferring sparse latent representations of documents/objeccts?* (2) *Is it possible to directly trade off sparsity against inference quality and inference time?* The first question has been addressed in Machine

[11] Webspam was retrieved from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

Learning. Existing regularization techniques can help us find sparse solutions, but cannot provide an affirmative answer to the second question. Our work provides a positive answer for both questions, at least for Topic Modeling, by realizing that the Frank-Wolfe algorithm for sparse approximation can help. Hence, it opens various potential directions for future research.

Acknowledgement. We would like to thank the reviewers for very helpful comments.

References

[1] Smola, A., Narayanamurthy, S.: An architecture for parallel topic models. Proceedings of the VLDB Endowment 3(1-2), 703–710 (2010)
[2] Hoffman, M.D., Blei, D.M., Bach, F.: Online learning for latent dirichlet allocation. In: NIPS, vol. 23, pp. 856–864 (2010)
[3] Newman, D., Asuncion, A., Smyth, P., Welling, M.: Distributed algorithms for topic models. The Journal of Machine Learning Research 10, 1801–1828 (2009)
[4] Asuncion, A.U., Smyth, P., Welling, M.: Asynchronous distributed estimation of topic models for document analysis. Statistical Methodology 8(1), 3–17 (2011)
[5] Wang, Q., Xu, J., Li, H., Craswell, N.: Regularized latent semantic indexing. In: SIGIR 2011, pp. 685–694. ACM (2011)
[6] Wang, Y., Bai, H., Stanton, M., Chen, W.-Y., Chang, E.Y.: PLDA: Parallel Latent Dirichlet Allocation for Large-Scale Applications. In: Goldberg, A.V., Zhou, Y. (eds.) AAIM 2009. LNCS, vol. 5564, pp. 301–314. Springer, Heidelberg (2009)
[7] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research 3(3), 993–1022 (2003)
[8] Sontag, D., Roy, D.M.: Complexity of inference in latent dirichlet allocation. In: Advances in Neural Information Processing Systems, NIPS (2011)
[9] Shashanka, M., Raj, B., Smaragdis, P.: Sparse overcomplete latent variable decomposition of counts data. In: NIPS (2007)
[10] Zhu, J., Xing, E.P.: Sparse topical coding. In: UAI (2011)
[11] Williamson, S., Wang, C., Heller, K.A., Blei, D.M.: The ibp compound dirichlet process and its application to focused topic modeling. In: ICML (2010)
[12] Wang, C., Blei, D.M.: Decoupling sparsity and smoothness in the discrete hierarchical dirichlet process. In: NIPS, vol. 22, pp. 1982–1989 (2009)
[13] Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. Machine Learning 42, 177–196 (2001)
[14] Clarkson, K.L.: Coresets, sparse greedy approximation, and the frank-wolfe algorithm. ACM Trans. Algorithms 6, 63:1–63:30 (2010)
[15] Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical Programming 103(1), 127–152 (2005)
[16] Lan, G.: An optimal method for stochastic composite optimization. Mathematical Programming, 1–33 (2011)
[17] Murray, W., Gill, P., Wright, M.: Practical optimization. Academic Press (1981)
[18] Forster, M.R.: Key concepts in model selection: Performance and generalizability. Journal of Mathematical Psychology 44(1), 205–231 (2000)
[19] Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: Liblinear: A library for large linear classification. Journal of Machine Learning Research 9, 1871–1874 (2008)
[20] Yu, H.F., Hsieh, C.J., Chang, K.W., Lin, C.J.: Large linear classification when data cannot fit in memory. ACM Trans. Knowl. Discov. Data 5(4), 23:1–23:23 (2012)

Learning Compact Class Codes for Fast Inference in Large Multi Class Classification

M. Cissé, T. Artières, and Patrick Gallinari

Laboratoire d'Informatique de Paris 6 (LIP6), Université Pierre et Marie Curie, Paris, France
firstname.lastname@lip6.fr,
http://www-connex.lip6.fr

Abstract. We describe a new approach for classification with a very large number of classes where we assume some class similarity information is available, e.g. through a hierarchical organization. The proposed method learns a compact binary code using such an existing similarity information defined on classes. Binary classifiers are then trained using this code and decoding is performed using a simple nearest neighbor rule. This strategy, related to Error Correcting Output Codes methods, is shown to perform similarly or better than the standard and efficient one-vs-all approach, with much lower inference complexity.

1 Introduction

Classification problems with very large number of classes (VLC) now occur in many applications in the web, text, image or video domains. Current problems often deal with tens or hundreds of thousand of classes. For example, for patent classification the number of classes is around 60 000, for image annotation classes are keywords and their number is not limited, the number of classes in large class hierarchies like Dmoz is around 600 000 and still growing.

Scaling algorithms for VLC is a recent research direction compared to scaling wrt the sample size or data dimensionality and this is still a challenging problem [1], [2] [3], [4], [5]. Its specificity lies in the complexity of inference. The inference linear complexity in the number of classes of standard one vs rest approaches is prohibitive for VLC and only sub-linear inference methods are acceptable for practical purpose. Of course, training should also remain feasible. Besides pure scaling problems, classes in VLC problems may evolve, e.g. some classes may become rarely observed. Designing classifiers that do not require full retraining for new classes is also important in many cases.

We focus here on the design of algorithms for dealing with these different issues. In our approach the classification problem is casted into a cost-sensitive framework where a class distance or class similarity information is supposed available. Cost sensitivity reflects an existing latent structure between the classes and these relations will be exploited as complementary knowledge to improve classification performance and to reduce the inference and training complexities.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 506–520, 2012.

This information could be provided by existing resources which in our case is a class-taxonomy, but the extension to other class similarity measures is straightforward.

Within this framework, the approach we develop relies on first learning binary class codes using the similarity information between classes, a class will then be represented as a l-dimensional binary code with values in $\{-1, +1\}$, and second in training l binary classifiers, each will predict one bit of the class code. The *dichotomizer* for the j^{th} bit of the code will be trained to distinguish between the samples of all classes whose j^{th} bit is 1 and those whose j^{th} bit is -1. A test example will then be categorized according to a simple nearest neighbor rule between the code computed for this example and learned codes. This method is inspired by Error Correcting Output Codes (ECOC) [6] and Class embeddings [7]. With this strategy, the complexity of inference will become linear in the length of the code instead of the number of classes for computing the output code of an input sample and logarithmic in the number of classes to compute the closest class code. Consequently we aim at designing compact class codes. Besides fast decoding, these codes should be discriminant enough to reach a performance equivalent to or higher than standard classification methods, at a reduced inference complexity.

Our main contribution is an efficient procedure for learning compact binary class codes of length l such that $l << k$ where k stands for the number of classes. The inference requires then computing the output of l classifiers while for the one vs rest (OVR) approach inference requires computing the output of k classifiers. The value of l may be set so as to achieve a compromise between complexity and acuracy. We show experimentally that the value of l required for reaching OVR performance, scales sub-linearly with the number of classes k and that increasing the complexity of the method (i.e. l) allows outperforming OVR. We provide an experimental comparison, with respect to performance and runtimes, of our method with baselines, including OVR, on datasets up to 10 000 classes built from the 2010 Large Scale Hierarchical Text Classification challenge datasets [8].

Finally, beyond its raw performance, we investigate the particular ability of our method for zero-shot learning, i.e. recognizing samples from new classes without any training sample. We show that providing the similarity information for new classes allows recognizing samples from theses classes even in the case when no training samples are available.

The paper is structured as follows. Section 2 reviews related works, section 3 presents our approach for learning compact class codes, and finally section 4 reports experimental results.

2 Related Works

Classification in a large number of classes has received an increasing attention in the last few years. The challenge of designing sub-linear inference complexity algorithms has guided the researchers into two main directions: hierarchical approaches and class nearest neighbor search methods.

Hierarchical Approaches exploiting a tree structured relation among classes are straightforward solutions for reducing the inference complexity from $O(k)$ to $O(\log k)$. Besides, many problems can be formulated as hierarchical classification, and most of the datasets available to the research community are organized hierarchically. Different methods have been proposed and some start from an existing hierarchy while others learn the class hierarchy. The filter tree and the conditional probability tree [9] for example are consistent reduction of multi-class problems to binary that learn a tree of classifiers. Trees offer a natural and efficient solution to the inference complexity problem, on the other hand, it is widely recognized (e.g. [3], [2]), that classifier cascades greatly suffer from the propagation of errors from parent to children. This is why some authors [10], [3] propose to globally train the classifiers in the tree, instead of using local classifiers, and report improved performance at the cost of larger training complexity.

The One-vs-Rest [11] approach is the most popular flat multi-class classifier. Surprisingly, it remains one of the most efficient approaches in terms of accuracy, for VLC [3]. Although the inference complexity is $O(k)$, it is readily parallellizable which might be another way for solving the complexity issue. The one-vs-rest classifier is then a strong contender for large scale classification and often the best classifier for VLC in terms of accuracy.

Taxonomy [7] and Label Embedding [3] are other flat approaches that propose to jointly learn a projection of the data and the classes (or the taxonomy) in a low dimensional latent space where each data will be close to its class representation. The inference procedure is based on a class nearest neighbor search, so that its complexity is potentially $O(\log k)$. This, and the competitive performance reported make these methods appealing for large multi-class problems though their performance is often below that of OVR method (e.g. [3]).

Error Correcting Output Coding [6] has not been used up to now for VLC. Since our method produces ECOCs, we introduce its principle here and will compare our strategy with standard ECOC in the experimental section. ECOC is a general framework for handling multi-class problems and consists in representing each class with a codeword. These codewords are arranged into a coding matrix $M(k \times l)$ where l is the code length and k is the number of classes. ECOC uses a binary coding $M \in \{-1, 1\}^{k \times l}$, each column of the coding matrix defines a partition of the target space and learning consists in training l dichotomizers to predict a codeword for each new instance. Prediction, also called decoding, is done by assigning a new sample to the class having the closest codeword according to a distance measure. The key issue here is designing a coding matrix with good error correcting properties. It is usually required that both rows and columns are well separated. Row separation ensures that a large number of binary classifiers have to make a wrong decision before the decoding process misclassifies a test sample. Column separation ensures that the binary dichotomizers (there is one dichotomizer per column) are uncorrelated. The most popular way for initializing M is to choose each m_{ij} to be 1 or -1 with probability $1/2$ [12], it is called dense random ECOC. For small number of classes, ECOC might

outperform the standard one-vs-rest scheme [6] [12] and the inference complexity is $O(\log k)$.

3 Our Approach: Learned Distributed Representation (LDR)

3.1 Principle

As demonstrated in recent publications one of the best performing method for VLC today is the OVR method [3]. Yet this strategy has inference complexity that scales linearly with the number of classes. Alternatively hierarchical methods allow fast inference but fail to reach the accuracy of OVR, due to error propagation in the tree. We aim here at building a method that allows, both fast inference and high accuracy. To reach this goal we propose a method called Learned Distributed Representation (LDR) that first learns binary low dimensional class codes, then uses binary classifiers to learn each bit of the codes, as in ECOC.

A key issue is to take into account the available relationships between classes (e.g. a hierarchical or a graph organization of classes). We propose to compute low dimensional binary class codes that reflect these relationships. In order to do that we first represent a class as a vector of similarities between the class and all other classes, $\mathbf{s}_i = [s(C_i, C_1), ..., s(C_i, C_k)]$ (see section 4 for an example). Different similarity measures may be used. It may be computed from a hierarchy of classes or from a similarity between samples of the two classes. Then, we learn short class codes that reflect these relationships between classes, by transforming these high k-dimensional representations of classes ($\mathbf{s}_i$) into lower l-dimensional codes ($\mathbf{h}_i$) via a dimension reduction algorithm. This step is explained in details in section 3.2. Once low dimensional (say l-dimensional, with $l << k$) binary class representations are learned, we train l binary classifiers, one for every bit. The binary classifier for the j^{th} bit is a dichotomizer that is learned to separate samples of all classes whose class code has the j^{th} bit set to 1 from the samples of all classes whose class code has the j^{th} bit set to -1. All these binary classifiers are then learned with all training samples from all classes.

Finally at test time, when one wants to decide the class of an input sample x, we use the l classifiers on x to compute a l-length binary word $\mathbf{m} = (m_1, ..., m_l)$ which is compared to the k class codes $\{\mathbf{h}_i, i = 1..k\}$ to find the nearest neighbor.

3.2 Learning Compact Binary Class-Codes

We propose to learn compact class codes with autoencoders which have been widely used for feature extraction and dimensionality reduction [13], [14]. Among many existing dimension reduction methods the advantage of autoencoders lies in the flexibility of the optimization criterion that allows us including additional terms related to class codes separation. An autoencoder is trained by minimizing a squared reconstruction error between the input (here a class representation $\mathbf{s}_i$)

and its reconstruction at the output of the autoencoder, $\widehat{\mathbf{s}}_i$. It may be viewed as an encoder (input $\rightarrow$ hidden layer) followed by a decoder (hidden $\rightarrow$ output layer). Usually it is required that encoding and decoding weights are tied [14], both for linear and non linear encoders, so that if $\mathcal{W}$ is the coding matrix, $\mathcal{W}^T$ is the decoding matrix. We used this strategy here. Training an autoencoder writes (omitting bias terms):

$$\underset{W}{\operatorname{argmin}} \sum_{i=1}^{k} ||\mathbf{s}_i - \mathcal{W}^T \times f(\mathcal{W} \times \mathbf{s}_i)||^2 \tag{1}$$

where $||.||$ is the euclidean distance. The activation function in hidden units f may be a linear function, then the projection learned by the autoencoder is similar to the one learned by a principal component analysis. One can expect to learn more interesting features by using nonlinearities on hidden units, using sigmoid or hyperbolic tangent activation functions (in our implementation, we use hyperbolic tangent activation function hidden units). To perform dimensionality reduction one uses a narrow hidden layer which forces to learn non trivial regularities from the inputs, hence interesting and compact codes on the hidden layer. The vector of activation of hidden units is the learned encoding function. Here the new class code for class C_i is then $\mathbf{h}_i = f(\mathcal{W} \times \mathbf{s}_i)$.

Ideally, new class codes should satisfy two properties. First, similar classes (according to the cost-sensitive information and/or to similar examples) should have close codes $\mathbf{h}_i$. Second, class codes for any pair of classes should be significantly different to ensure accurate classification at the end. The first property is naturally satisfied since an autoencoder actually learns hidden codes that preserve distances in the original space. Next, to ensure minimal separation between class codes we propose to look for a solution of the following constrained problem:

$$\begin{aligned} &\underset{\mathcal{W}}{\operatorname{argmin}} \sum_{i=1}^{k} ||\mathbf{s}_i - \mathcal{W}^T \times f(\mathcal{W} \times \mathbf{s}_i)||^2 \\ &\text{s.t. } \forall (i,j), i \neq j : ||f(\mathcal{W} \times \mathbf{s}_i) - f(\mathcal{W} \times \mathbf{s}_j)|| \geq b \end{aligned} \tag{2}$$

The constraints are inspired from margin based learning and yield to maximize the distance between any pair of class codes up to a given threshold b. We solve this optimization problem by stochastic gradient descent using the unconstrained regularized form:

$$\begin{aligned} \underset{\mathcal{W}}{\operatorname{argmin}}\ \alpha &\sum_{i=1}^{k} ||\mathbf{s}_i - \mathcal{W}^T \times f(\mathcal{W} \times \mathbf{s}_i)||^2 \\ &+ \beta \sum_{i,j=1}^{k} \max(0, b - ||f(\mathcal{W} \times \mathbf{s}_i) - f(\mathcal{W} \times \mathbf{s}_j)||) \\ &+ \frac{\lambda}{2} ||\mathcal{W}||^2 \end{aligned} \tag{3}$$

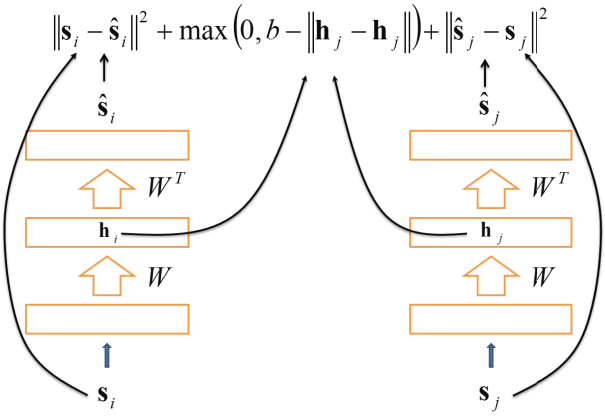


Fig. 1. Learning the autoencoder from pairs of input samples (here α and β are considered equal to 1). See Algorithm 1 for details.

where α and β weight the respective importance of the reconstruction error term and of the margin terms, and $||\mathcal{W}||^2$ is a regularization term. Note that α, β, and b (which tunes the margin between two class codes) are set by cross validation.

We learn the autoencoder using stochastic gradient descent by iteratively picking two training samples i and j at random and making a gradient step. Figure 1 illustrates the training process which recalls somehow Siamese architectures used in the past for vision tasks [15]. At the end, in order to get binary class codes, we threshold the learned real valued class codes. This means that the j^{th} component of all class codes $\mathbf{h}_i$ are set to $\mathbf{h}_i(j) = -1$ if $\mathbf{h}_i(j) < \theta_j$, and $\mathbf{h}_i(j) = +1$ otherwise. The threshold value θ_j is chosen so that the prior probability of the j^{th} bit of a class code be $+1$ is equal to 0.5, and this is done by setting θ_j to the median of $\{\mathbf{h}_i(j)|i = 1...k\}$. Although this cut-off it is not learned to optimize classification accuracy, it should be noted that it is defined according to the usual property in ECOC (firing with probability 0.5). Also since similar classes should have close class codes, it is expected that the obtained two class classification problem (i.e. for the j^{th} bit of class codes, separating samples of all classes with $\mathbf{h}_i(j) = +1$ from the samples of all classes with $\mathbf{h}_i(j) = -1$) should be easier to solve than any random two class problem as those defined in traditional ECOC. We will come back to this point in the next section. Algorithm 1 describes the whole algorithm.

3.3 Relations to ECOC

Because each element in the class codes has probability 1/2 of being either $+1$ or -1, our method bares some similarities with the standard dense random ECOC. However, there are two fundamental differences.

The first difference is that by construction, our learned distributed representation is intended to have a reduced tree induced loss compared to randomly generated methods because the autoencoder projects classes that are close in the hierarchy in the same area of the latent space. The second difference, which is somehow related to the first one, is that the binary classification problems induced by the learned class codes should be easier than in random ECOC. Indeed, since similar classes should have close class codes, it is likely that for similar classes most bits are equal. This means that a particular dichotomizer is trained with samples for class +1 and for class -1 that are more homogeneous than if the partitioning of classes was random, as in traditional ECOCs. At the end, if dichotomizers reach higher accuracy, the overall accuracy of the multiclass classifier should also be higher.

Algorithm 1. Learning Compact Binary Class Codes

Input: $\{\mathbf{s}_i \in \mathbb{R}^k | i = 1, ...k\}$, l, ϵ
Output: $\{\mathbf{h}_i \in \mathbb{R}^l | i = 1, ...k\}$

Learn the weights $\mathcal{W}$ of an autoencoder (with k input neurons, l hidden neurons, and k output neurons) on $\{\mathbf{s}_i \in \mathbb{R}^k | i = 1, ...k\}$ to minimize cost in Eq. (3)
repeat
  Pick randomly two samples $(\mathbf{s}_i, \mathbf{s}_j)$
  Make a gradient step : $\mathcal{W} = \mathcal{W} - \epsilon \partial L_{\mathcal{W}}(\mathbf{s}_i, \mathbf{s}_j)/\partial \mathcal{W}$
  with: $L_{\mathcal{W}}(\mathbf{s}_i, \mathbf{s}_j) = \frac{1}{2} \sum_{k \in \{i,j\}} \alpha ||\mathbf{s}_k - \mathcal{W}^T \times f(\mathcal{W} \times \mathbf{s}_k)||^2 + \lambda ||\mathcal{W}||^2 + \beta \max(0, b - ||f(\mathcal{W} \times \mathbf{s}_i) - f(\mathcal{W} \times \mathbf{s}_j)||)$
until convergence criterion is met
Compute the learned class codes $\forall i \in [1, k], \mathbf{h}_i = f(\mathcal{W} \times \mathbf{s}_i)$
for all $j = 1...l$ **do**
  Compute the median θ_j of the j^{th} component of $\mathbf{h}_i$'s, $\{\mathbf{h}_i(j) | i = 1, ..., k\}$
  Threshold the j^th component of $\mathbf{h}_i$'s at θ_j so that $\forall i \in [1, k], \mathbf{h}_i(j) = \begin{cases} 1 \text{ if } \mathbf{h}_i(j) \leq \theta \\ -1 \text{ otherwise} \end{cases}$
end for
return Compact binary class codes $\{\mathbf{h}_i \in \mathbb{R}^l | i = 1, ...k\}$

An ECOC coding scheme closer to our method is the discriminative ECOC (DECOC) which learns a discriminative coding matrix by hierarchically partitioning the classes according to a discriminative criteria [16]. The hierarchy is built so as to maximize the mutual information between the data in each partition and the corresponding labels. Our method differs from this in that we are seeking codewords having a sub-linear dependency on the number of classes k while the DECOC method creates codewords of length $k - 1$.

3.4 Training and Inference Complexity

We focus here on complexity issues with respect to the number of classes k, the number of training samples N, the dimension of samples d, and the length of

the learned class codes l. Let us denote by $C_T(N)$ the complexity of training one binary classifier with N training samples, and by C_I the complexity of inference for a binary classifier. All complexities in the following will be expressed as a function of C_T and C_I.

We start with our method. Training consists in learning the class codes of length l, then in learning l classifiers. Learning class codes is done by gradient descent whose complexity depends on the number of iterations. Yet since class codes are binarized at the end, one can expect that the method will not be very sensitive to accurate convergence of the autoencoder and one can reasonably assume a fixed and limited number of iterations $\mathcal{I}$ so that learning the autoencoder requires $O(\mathcal{I} \times k^2 \times l)$ (I iterations with k samples every iteration whose forward and backward pass costs roughly $O(k \times l)$). Next, learning the l binary classifiers requires $O(l \times C_T(N))$. At the end training complexity is in $O(\mathcal{I} \times k^2 \times l + l \times C_T(N))$. Inference consists in finding the class code which is most similar (wrt. Hamming distance) to the output code computed for this input sample. Computing the output code requires using the l classifiers, hence $O(l \times C_I)$. Next, using fast nearest neighbor search methods such as ball trees or kd-trees for finding the closest class code may be done (in practice) in $O(\log k)$ comparisons [17], where each comparison costs $O(l)$. Overall, the inference complexity is then $O(l \times (\log k + C_I))$.

We compare these costs to those of the OVR method which is the most accurate technique for large scale classification [3] (see Table 1). Training in OVR method requires $O(k \times C_T(N))$ since one uses k classifiers that are all trained with all training samples, while inference requires $O(k \times C_I)$.

It clearly appears from this discussion that OVR does not extend easily to VLC due to its inference complexity that scales linearly with the number of classes. Compared to these baselines, our method exhibits interesting features. As we will argue from experimental results, it may outperform OVR for $l << k$ and the minimal length l for such a behavior seems to scale strongly sublinearly with k. Furthermore although the training complexity includes a term in $O(k^2)$, it must be clear that in experimental settings such as the ones we investigate in this paper (large number of samples and high dimensionality), the overall training complexity in $O\left(l\mathcal{I}k^2 + lC_T(N)\right)$ is dominated by the second term $O(lC_T(N))$.

Table 1. Comparison of training and inference complexity for our method and for standard methods, OVR and ECOC, as a function of the number of classes k, the dimension of the data d, the size of the class codes l, the learning complexity of a binary classifier with N training samples $C_T(N)$, the inference complexity of a binary classifier C_I, and the number of training iterations I of the autoencoder (LDR method).

	Training	**Inference**
OVR	$O(kC_T(N))$	$O(kC_I)$
ECOC(l)	$O(lC_T(N))$	$O\left(lC_I + l\log k\right))$
LDR(l)	$O\left(l\mathcal{I}k^2 + lC_T(N)\right)$	$O\left(lC_I + l\log k\right)$

4 Experiments

We performed experiments on three large scale multi-class single label datasets. The proposed method (LDR) is compared to two coding methods, spectral embedding (SPE) and traditional error correcting output coding (ECOC), and to a standard OVR baseline. We first present the datasets, then we explain our experimental setup and finally we present results and analysis.

4.1 Datasets

We used datasets with respectively 1000, 5000 and 10000 classes. Each dataset was created by randomly selecting the corresponding classes from a large scale dataset released for the first PASCAL large scale hierarchical text classification challenge (Kosmopoulos et al., 2010). This dataset was extracted from the open Mozilla directory DMOZ (www.dmoz.org). The classes are organized in a tree hierarchy, classes being at the leaves of the hierarchy and internal nodes being not instantiated classes. Hierarchies are of depth 5 (Kosmopoulos et al., 2010).

The documents were provided as word counts, and then transformed into normalized TF/IDF feature vectors. Considering that for large multi-class text classification every new class is likely to bring specific new words, we did not performed any feature selection although all datasets have very high dimensional feature spaces.

Statistics of the datasets are detailed in Table 2. Each dataset is split into training, validation and testing sets (see Table 2).

We exploited a similarity measure between classes i and j, which is defined as a function of the distance $d_{i,j}$ between the two classes in the hierarchy measured by the length of the shortest path in the tree between the two classes: $\mathbf{s}_i(j) = s(C_i, C_j) = \exp(-d_{i,j}^2/2\sigma^2)$. The tree path distance between two classes is also used in the tree loss used as a classification measure in section 4.3. We systematically used $\sigma = 1$ in our experiments.

Table 2. Statistics of the dataset used in the experiments

Statistics	10^3 classes	$5 * 10^3$ classes	10^4 classes
Nb. training docs	8119	36926	76417
Nb. validation docs	3005	13855	28443
Nb. testing docs	3006	13771	28387
Nb. features	347 255	-	-

4.2 Experimental Setup

Three classifiers were used as baselines: OVR, random ECOC and a Spectral Embedding technique.

Besides ECOC classifiers, we also compared our method to a spectral embedding technique (SPE) which can be used for learning class codes from a similarity matrix and is an alternative to our auto-associator method. Spectral embedding is widely used as a preprocessing step before applying k-means in clustering applications. It has also been used recently for hashing and we exploit a similar idea here. In [18] the authors propose to embed the data for fast retrieval by binarizing the components of the eigenvectors of the similarity matrix Laplacian. This process aims at mapping similar examples in the same regions of a target space. The training complexity of the method is $O(k^3 + lC_T(N))$, which is much larger than LDR or ECOC, and is due to the high complexity if the eigen-decomposition. This method is similar in spirit to LDR and ECOC and is a natural candidate for comparison. The classes here play the same role as data do in spectral hashing.

We chose logistic regression as a base classifier (dichotomizers) for all methods, but any other binary classifier could be used as well. The binary classifiers were trained with a regularization parameter selected from $\lambda \in \{0.001, 0.0005, ..., 10^{-6}\}$ using the validation set.

To train random ECOC classifiers, for a given code length l and a number of class k, we generated several $k \times l$ matrices and discarded those having equal or complementary rows. We then used the coding matrices with best error correcting property (the top 25 matrices for 10^3 classes and the top 10 for $5 * 10^3$ and 10^4 classes) to train an ECOC classifier. Then we kept the model that reached the best performance on the validation set for evaluation on the test set.

We compare the methods using accuracy and tree induced loss which is defined as the average of the length of the shortest path in the hierarchy between the correct class and the predicted class. The tree induced loss measures the ability of the classifier to take into account the hierarchical nature of the classification problem, and the class proximity according to this metric. A low tree loss means that confusions are made between neighboring classes, while a high tree loss signifies that confusions occur among distant classes.

4.3 Comparison of the Methods

We investigate here the behavior of the different methods on the three datasets and explore how the performance evolves with respect to the class code length. Comparisons with all methods are performed on the 10^3 and $5 * 10^3$ classes corpora, while on the larger 10^4 classes dataset, only OVR vs LDR were tested. Figure 2 reports accuracies on the first two datasets for code length in $\{200, 300, 400, 500, 600\}$. First it can be seen that LDR outperforms systematically the two other coding methods (SPE and ECOC) whatever the dataset, and whatever the class code length. Second, the performance of the three coding methods (LDR, SPE and ECOC) increases, with some fluctuation, with the code length. A higher code is needed when the number of classes increases. This behavior is intuitive. Finally one can see that LDR reaches and even exceeds the performance of OVR on these two datasets, while ECOC and SPE stay under the performance of OVR, even when increasing the code length l.

Table 3 compares the different methods using their best accuracy score[1], and the corresponding tree induced loss on the same two datasets. It can be seen that the best performance of the different methods are quite close, LDR being systematically higher and providing a clear speedup wrt OVR. For example, for 1 000 classes, with a code length of 200 LDR achieves an accuracy of 67.49% while OVR's accuracy is 66.50%. In this case, the number of classifiers used by the OVR method is 5 times that of LDR.

We come back to our previous observation that LDR is consistently better than random error correcting output coding (ECOC) (Figure 2), which holds whatever the code length. Our main explanation of this phenomenon is that the binary problems are probably easier to solve with LDR. It has been observed since the early use of ECOCs [6] that the dichotomies induced by the codes where more difficult to solve than the initial OVR dichotomies. Here, neighbor classes in the tree, are forced to have similar codes. The data for these classes are often closer one to the other than that of distant classes, so that similar inputs will most often be required to be classified similarly. On the opposite, classical ECOCs where codes are designed at random do not share this property. To investigate this, we compared the mean accuracy of the binary classifiers induced by our method to the mean accuracy of classifiers in a random ECOC scheme. The mean accuracy remains between 72% and 75% for LDR while it is constant at about 69% for ECOC which confirms the hypothesis that learned dichotomizers induce easier problems. Also we think that the learning criteria of the autoencoder helps creating better class codes than those produced by the spectral embedding method.

At last we compare LDR and OVR on classification tasks with up to 10 000 classes. Figure 3 shows the performance of LDR vs OVR for the three datasets ($10^3, 5 * 10^3$ and 10^4 classes) for a code length of size 500. LDR outperforms OVR whatever the number of classes. Speedup are more and more important as the number of classes increases. For 10^4 classes LDR achieves an accuracy of 36.81% (with a code length of 500) while the OVR's performance is 35.20%. This performance is achieved while using 20 times less classifiers than the number of classes. This corresponds to a speedup of 46 wrt OVR (measured by runtimes). Such a speedup is not only due to the smaller number of classifiers used by LDR, but also to fast bitcounts routines that exploit the binary representation of codes for nearest neighbour search.

4.4 Zero-Shot Learning

A few approaches have been proposed in the literature to answer the *zero-shot learning* problem [19], [20], i.e. designing a classifier that is able to discriminate between classes for which we do not have instances in the training set. One particular approach proposes the use of a rich semantic encoding of the classes [20]. Our approach is close to this idea since the codes of classes

[1] For each method, one uses the parameterization, including the value of l, leading to the best score.

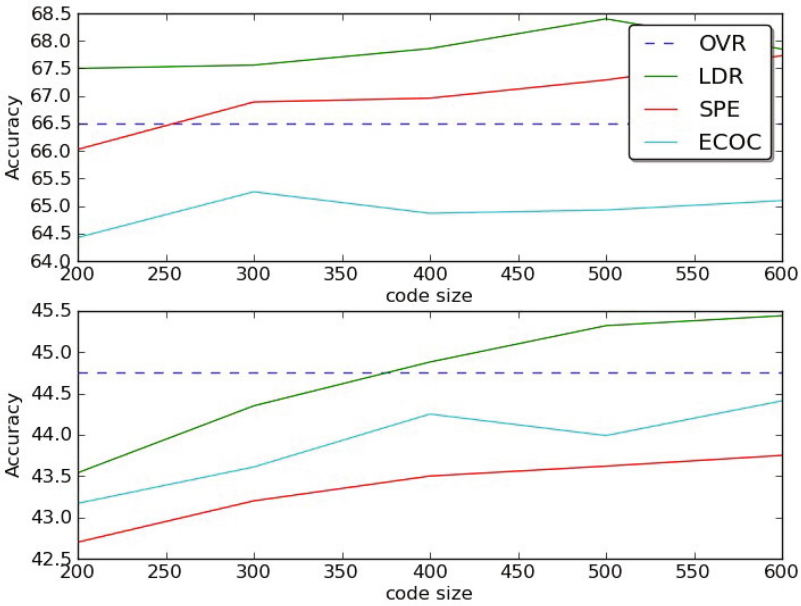


Fig. 2. Accuracy of our method (LDR), random ECOC (ECOC), Spectral Embedding (SPE), and OVR as a function of code length on datasets with 1 000 classes (top) and with 5 000 classes

Table 3. Comparative results of OVR, Random ECOC, Spectral Embedding, and LDR, on datasets with 1000 and 5000 classes with respect to accuracy, tree induced loss, and inference runtime. The runtimes are given as speed-up factors compared to OVR (×2 means twice as fast as OVR). Reported results are the best ones obtained on the datasets whatever the class code length. For LDR, we also provide the performance reached for a minimal l yielding performance at least equal to that of OVR, denoted as LDR (first), to to stress the speed-up.

Classifiers	1000 classes			5000 classes		
	Accuracy	T.I.L	Speed	Accuracy	T.I.L	Speed
One-vs-rest	66.50%	2.63	×1	44.76%	3.98	×1
Random ECOC	65.10%	2.74	×2	44.41%	4.12	×12
SPE	67.73%	2.51	×2	43.75%	4.30	×12
LDR (first)	67.49%	2.54	×5	44.88%	3.98	×17
LDR(best)	**68.40%**	**2.46**	×2	**45.44%**	**3.93**	×12

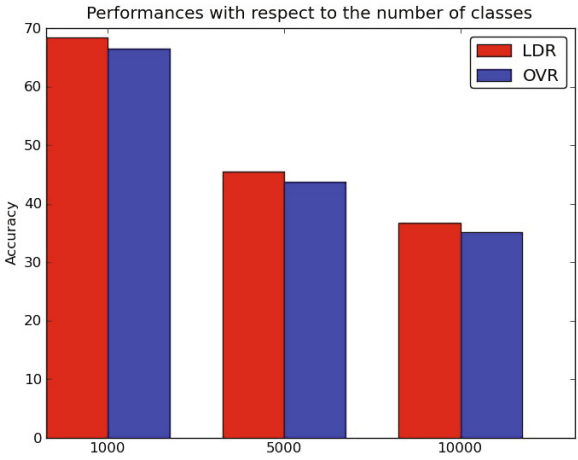


Fig. 3. Accuracy of our method (LDR) and OVR on datasets with 1 000, 5 000 and 10 000 classes. Whatever the dataset LDR exploits class codes of length $l = 500$.

Table 4. Average accuracy (and standard deviation) of LDR ($l = 200$) for zero-shot learning tasks. Results are averaged over 10 runs with removal of different random sets of classes.

# classes removed	10	20	30	40	50
Accuracy (std)	25.64(12.20)	24.45(6.34)	16.76(4.24)	14.31(3.18)	12.76(2.48)

(computed by the autoencoder) are vectors that encode some semantic information on classes.

To explore empirically how our model is able to achieve zero-shot learning, we performed the following experiment on the 1000 classes dataset. We learned the class codes on the 1000 class representations (similarity vectors) computed from the hierarchy, s_i. Then we selected randomly a number of classes (10 to 50) and removed all training samples of these classes from the training set. The dichotomizers were then trained with this reduced training set. At test time, following the approach in [19], we use the learned classifier to discriminate between the classes whose training samples were not present in the training set. Results are given in Table 4 for a class code length equal to 200. One can see that the accuracy achieved by LDR on classes that have not been learned is significantly greater than a random guess although it is naturally lower than the accuracy obtained on classes that were actually represented in the training set as reported in previous section.

Note also that one could go one step further than the zero-shot paradigm and try to recognize samples from a new class which was even not used for learning

the class codes, provided one gets its similarity with all classes in the training stage. This would fit with many large multi-class problems where the set of classes is not closed (for instance new classes appear periodically in the DMOZ repository). Preliminary results show a similar performance as above provided the number of new classes remains small. This is a perspective of our work.

5 Conclusion

We have presented a new approach for dealing with hierarchical classification in a large number of classes. It combines the accuracy of flat methods and the fast inference of hierarchical methods. It relies on building distributed compact binary class codes that preserve class similarities. The main features of the method lies in its inference complexity that scales sub-linearly with the number of classes while outperforming the standard OVR and Error Correcting Output Codes techniques on problems up to 10 000 classes. Interestingly our approach also allows, to some extent, considering the addition of new classes in the hierarchy without providing training samples, an instance of the zero-shot learning problem.

References

1. Weinberger, K., Chapelle, O.: Large margin taxonomy embedding for document categorization. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 21, pp. 1737–1744 (2009)
2. Bennett, P.N., Nguyen, N.: Refined experts: improving classification in large taxonomies. In: SIGIR, pp. 11–18 (2009)
3. Bengio, S., Weston, J., Grangier, D.: Label embedding trees for large multi class tasks. In: Advances in Neural information Processing (2010)
4. Xiao, L., Zhou, D., Wu, M.: Hierarchical classification via orthogonal transfer. In: Getoor, L., Scheffer, T. (eds.) Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 801–808. ACM, New York (2011)
5. Deng, J., Satheesh, S., Berg, A.C., Li, F.F.: Fast and balanced: Efficient label tree learning for large scale object recognition. In: NIPS, pp. 567–575 (2011)
6. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2, 263–286 (1995)
7. Weinberger, K., Chapelle, O.: Large taxonomy embedding with an application to document categorization. In: Advances in Neural Information Processing (2008)
8. Kosmopoulos, A., Gaussier, E., Paliouras, G., Aseervatham, S.: The ecir 2010 large scale hierarchical classification workshop. SIGIR Forum 44(1), 23–32 (2010)
9. Beygelzimer, A., Langford, J., Lifshits, Y., Sorkin, G., Strehl, A.: Conditional probability tree estimation analysis and algorithms. In: Proceedings of the Twenty-Fifth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI 2009), pp. 51–58. AUAI Press, Corvallis (2009)
10. Cai, L., Hofmann, T.: Hierarchical document categorization with support vector machines. In: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, pp. 78–87 (2004)

11. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. J. Mach. Learn. Res. 5, 101–141 (2004)
12. Allwein, E.L., Schapire, R.E., Singer, Y., Kaelbling, P.: Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research 1, 113–141 (2000)
13. Gallinari, P., LeCun, Y., Thiria, S., Fogelma-soulie, F.: Mémoires associatives distribuées: une comparaison (distributed associative memories: a comparison). In: Proceedings of COGNITIVA 1987, Paris, La Villette, Cesta-Afcet (May 1987)
14. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine learning, ICML 2008, pp. 1096–1103. ACM, New York (2008)
15. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a siamese time delay neural network. In: NIPS, pp. 737–744 (1993)
16. Pujol, O., Escalera, S., Radeva, P.: An incremental node embedding technique for error correcting output codes. Pattern Recogn. 41(2), 713–725 (2008)
17. Moore, A.: Efficient memory-based learning for robot control (October 1990)
18. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS, pp. 1753–1760 (2008)
19. Larochelle, H., Erhan, D., Bengio, Y.: Zero-data learning of new tasks. In: AAAI, pp. 646–651 (2008)
20. Palatucci, M., Pomerleau, D., Hinton, G.E., Mitchell, T.M.: Zero-shot learning with semantic output codes. In: NIPS, pp. 1410–1418 (2009)

ParCube: Sparse Parallelizable Tensor Decompositions

Evangelos E. Papalexakis[1,⋆], Christos Faloutsos[1], and Nicholas D. Sidiropoulos[2,⋆⋆]

[1] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
{epapalex,christos}@cs.cmu.edu
[2] Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA
nikos@ece.umn.edu

Abstract. How can we efficiently decompose a tensor into sparse factors, when the data does not fit in memory? Tensor decompositions have gained a steadily increasing popularity in data mining applications, however the current state-of-art decomposition algorithms operate on main memory and do not scale to truly large datasets. In this work, we propose ParCube, a new and highly parallelizable method for speeding up tensor decompositions that is well-suited to producing sparse approximations. Experiments with even moderately large data indicate over 90% sparser outputs and 14 times faster execution, with approximation error close to the current state of the art irrespective of computation and memory requirements. We provide theoretical guarantees for the algorithm's correctness and we experimentally validate our claims through extensive experiments, including four different real world datasets (Enron, Lbnl, Facebook and Nell), demonstrating its effectiveness for data mining practitioners. In particular, we are the first to analyze the very large Nell dataset using a sparse tensor decomposition, demonstrating that ParCube enables us to handle effectively and efficiently very large datasets.

Keywords: Tensors, PARAFAC decomposition, sparsity, sampling, randomized algorithms, parallel algorithms.

⋆ Research was sponsored by the Defense Threat Reduction Agency and was accomplished under contract No. HDTRA1-10-1-0120. Funding was provided by the U.S. Army Research Office (ARO) and Defense Advanced Research Projects Agency (DARPA) under Contract Number W911NF-11-C-0088. Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative 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 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. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

⋆⋆ N. Sidiropoulos was partially supported by ARO contract W911NF-11-1-0500.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 521–536, 2012.

1 Introduction

Tensors and tensor decompositions have recently attracted considerable attention in the data mining community. With the constantly increasing volume of today's multi-dimensional datasets, tensors are often the 'native' format in which data is stored, and tensor decompositions the natural modeling toolset - albeit still suffering from major scalability issues. The state of the art toolboxes for tensors [8,4] still operate on main memory and cannot possibly handle disk-resident tensor datasets, in the orders of millions or billions of non-zeros.

Motivated by the success of random sampling - based matrix algorithms such as [11], it is natural to ask whether we can we use similar tools in the case of tensors. Is it possible to randomly under-sample a tensor multiple times, process the different samples in parallel and cleverly combine the results at the end to obtain high approximation accuracy at low complexity and main memory costs? There exists important work on how to use sampling in order to achieve a sparse matrix decomposition, the CUR decomposition [11]; this method has also been extended in order to handle tensors [15]. However, both these methods are tied to a specific decomposition, while we desire to disconnect sampling from the specific decomposition that follows.

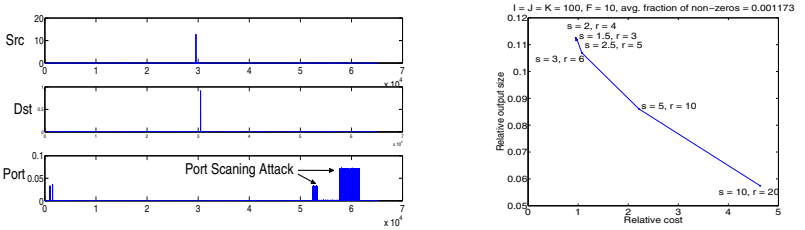
This paper introduces ParCube, a fast and parallelizable method for speeding up tensor decompositions by leveraging random sampling techniques. A nice side-benefit of our algorithm is its natural tendency to produce sparse outer-product approximations, i.e., the model-synthesized approximation of the given tensor data is naturally very sparse, which is a desirable property in many applications. Our core contribution is in terms of the merging algorithm that collects the different 'punctured' decompositions and combines them into one overall decomposition in an efficient way. We provide theoretical guarantees for the correctness of our approach.

In Fig. 1 we demonstrate a preview of our results: On subfigure 1(a), we show a successful discovery of what appears to be a *port scanning attack*, on the LBNL network traffic dataset, and subfigure 1(b) demonstrates over 90% sparser results than regular Parafac, while maintaining the same approximation error.

The rest of this paper is structured as follows. Section 2 provides some useful background; section 3 describes the proposed method, and section 4 contains experiments. Related work is reviewed in section 5, and conclusions are drawn in section 6.

2 Tensor Decompositions

A Note on Notation. A scalar is denoted by a lowercase, italic letter, e.g. x. A column vector is denoted by a lowercase, boldface letter, e.g. $\mathbf{x}$. A matrix is denoted by an uppercase, boldface letter, e.g. $\mathbf{X}$. A three-way tensor is denoted by $\underline{\mathbf{X}}$. Let $\mathcal{I}$ be a set of indices, e.g. $\mathcal{I} = \{1, 4, 7\}$; then, $\mathbf{a}(\mathcal{I})$ denotes $\{\mathbf{a}(1), \mathbf{a}(4), \mathbf{a}(7)\}$; $\mathbf{a}(:)$ spans all the elements of $\mathbf{a}$. This notation naturally extends to matrices and tensors, i.e., $\mathbf{A}(\mathcal{I}, :)$ comprises all columns of $\mathbf{A}$ restricted to rows in $\mathcal{I}$. By $NNZ(\ )$ we denote the number of non-zeros.



(a) Port Scanning Attack-like behaviour on the LBNL Network Traffic Dataset

(b) Over 90% sparser results than regular PARAFAC, with same approximation error

Fig. 1. A snapshot of our results. In (a) we have one source (addr. 29571) contacts one destination (addr. 30483) using a wide range of near consecutive ports and same amount of packets. In (b), we see that with same relative error, we achieve 90 % sparser outputs, compared to the ALS-PARAFAC algorithm.

Tensors. A tensor of n modes (or n-way/n-mode tensor) is a structure indexed by n variables. For example, a matrix is a two-way tensor. In this work, we focus on three-way tensors, because they are most common; however, all results can be readily extended to higher-way tensors. A three-way tensor $\underline{\mathbf{X}}$ is a structure that resembles a data cube. A detailed survey for tensors and tensor decompositions may be found in [14].

The PARAFAC Decomposition. The PARAFAC decomposition [12] of $\underline{\mathbf{X}}$ into F components is $\underline{\mathbf{X}} \approx \sum_{f=1}^{F} \mathbf{a}_f \circ \mathbf{b}_f \circ \mathbf{c}_f$, where $\mathbf{a} \circ \mathbf{b} \circ \mathbf{c}(i,j,k) = \mathbf{a}(i)\mathbf{b}(j)\mathbf{c}(k)$.

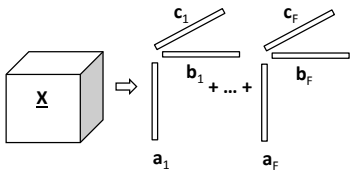


Fig. 2. The F-component PARAFAC decomposition of $\underline{\mathbf{X}}$

The most popular algorithm for fitting the PARAFAC decomposition is the Alternating Least Squares (ALS) [9,14]. The computational complexity of the ALS Algorithm for a $I \times J \times K$ tensor, and for F components is $O(IJKF)$ per iteration.

3 The ParCube Method

In this section we introduce PARCUBE, a new method for PARAFAC decomposition designed with three main goals in mind: **G1**: Relative simplicity, speed, and

parallelizable execution; **G2**: Ability to yield sparse latent factors and a sparse tensor approximation; and **G3**: provable correctness in merging partial results, under appropriate conditions.

Sampling for ParCube. The first step of ParCube is to sample a very high dimensional tensor and use the sampled tensor in lieu of the original one, bearing three important requirements in mind: **R1** The need to significantly reduce dimensionality; **R2** The desire that sampling should be decomposition-independent - we should be able to apply any decomposition we desire *after* sampling, and be able to extrapolate from that; and **R3**: Sampling should maintain linear complexity on the number of non-zero entries.

The first thing that comes to mind in order to satisfy requirement **R1** is to take a uniform random sample of the indices of each mode, i.e., take a uniform random sample of the index sets $\{1 \cdots I\}$, $\{1 \cdots J\}$, and $\{1 \cdots K\}$. However, this naive approach may not adequately preserve the data distribution, since the random index samples may correspond to entirely arbitrary rows/columns/fibers of the tensor. We performed initial tests using this naive method, and the results were consistently worse than the proposed method's. We thus propose to do *biased* sampling: If we, somehow, determine a measure of importance for each index of each mode, then we may sample the indices using this measure as a sampling weight/probability. For the purposes of this work, let us assume that our tensor $\underline{\mathbf{X}}$ has *non-negative* entries (which is the case in huge variety of data mining applications); if we were to deal with tensors containing real values, we should consider the element-wise absolute value of the tensor for the notions that we introduce in the sequel.

A reasonable measure of importance is the marginal sum of the tensor for each mode [1]. Namely, the measure of importance for the indices of the first mode is defined as: $\mathbf{x}_a(i) = \sum_{j=1}^{J} \sum_{k=1}^{K} \underline{\mathbf{X}}(i,j,k)$ for $i = 1 \cdots I$.

Similarly, we define the following importance measures for modes 2 and 3:

$$\mathbf{x}_b(j) = \sum_{i=1}^{I} \sum_{k=1}^{K} \underline{\mathbf{X}}(i,j,k), \mathbf{x}_c(k) = \sum_{i=1}^{I} \sum_{j=1}^{J} \underline{\mathbf{X}}(i,j,k)$$

for $j = 1 \cdots J$ and $k = 1 \cdots K$.

Intuitively, if $\mathbf{x}_a(i)$ is high for some i, then we would desire to select this specific index i for our sample with higher probability than others (which may have lower $\mathbf{x}_a$ value). This is the very idea behind ParCube: We sample the indices of each mode of $\underline{\mathbf{X}}$ without replacement, using $\mathbf{x}_a$, $\mathbf{x}_b$ and $\mathbf{x}_c$ to bias the sampling probabilities.

We define s to be the sampling factor, i.e. if $\underline{\mathbf{X}}$ is of size $I \times J \times K$, then $\underline{\mathbf{X}}_s$ derived by ParCube will be of size $\frac{I}{s} \times \frac{J}{s} \times \frac{K}{s}$. We may also use different sampling factors for each mode of the tensor, without loss of generality.

[1] Another, reasonable alternative is the sum-of-squares of the elements of rows, columns and fibers, which is a measure of *energy*. We leave this for future work.

In order to obtain the sample we 1) Compute set of indices $\mathcal{I}$ as random sample without replacement of $\{1 \cdots I\}$ of size I/s with probability $p_{\mathcal{I}}(i) = \mathbf{x}_a(i)/\sum_{i=1}^{I} \mathbf{x}_a(i)$. 2) Compute set of indices $\mathcal{J}$ as random sample without replacement of $\{1 \cdots J\}$ of size J/s with probability $p_{\mathcal{J}}(j) = \mathbf{x}_b(j)/\sum_{j=1}^{J} \mathbf{x}_b(j)$. 3) Compute set of indices $\mathcal{K}$ as random sample without replacement of $\{1 \cdots K\}$ of size K/s with probability $p_{\mathcal{K}}(k) = \mathbf{x}_c(k)/\sum_{k=1}^{K} \mathbf{x}_c(k)$.

The PARCUBE method defines a means of sampling the tensor across all three modes, without relying on a specific decomposition or a model. Therefore, it satisfies requirement **R3**. Algorithm 1 provides an outline of the sampling forPARCUBE.

Lemma 1. *The computational complexity of Algorithm 1 is linear in the number of non zero elements of* $\underline{\mathbf{X}}$.

Proof. Suppose we have a representation of $\underline{\mathbf{X}}$ in quadruplets of the form (i, j, k, v) where $\underline{\mathbf{X}}(i, j, k) = v$, for $v \neq 0$ and $v \in NNZ(\underline{\mathbf{X}})$. For each of these quadruplets, we may compute the density vectors as:

$$\mathbf{x}_a(i) = \mathbf{x}_a(i) + v, \mathbf{x}_b(j) = \mathbf{x}_b(j) + v, \mathbf{x}_c(k) = \mathbf{x}_c(k) + v$$

This procedure requires 3 $O(1)$ additions per element v, therefore the total running time is $O(NNZ(\underline{\mathbf{X}}))$. ■

By making use of the above Lemma, and noticing that sampling of the elements, after having computed the densities of each mode is a linear operation on the number of non-zeros, we conclude that requirement **R3** is met, i.e. our computation of the biases and biased sampling are linear on the number of non-zeros. Furthermore, sampling pertains to Goal **G1** which calls for a fast algorithm.

Non-negative PARAFAC Decomposition Using ParCube. Now, let us demonstrate how to apply PARCUBE in order to scale up the popular PARAFAC decomposition, with non-negativity constraints. We choose to operate under the non-negativity regime since the vast majority of applications of interest naturally impose this type of constraint.

Algorithm 2 demonstrates the most basic approach in which one extracts a sample from the original tensor, runs the PARAFAC decomposition on that (significantly) smaller tensor and then redistributes the factor vectors to their original positions, according to the sampled indices $\mathcal{I}, \mathcal{J}, \mathcal{K}$. Note that many of the coefficients of the resulting PARAFAC factor matrices will be exactly zero, since their corresponding indices will not be included in the sample and consequently, they will not receive an updated value. This implies that a natural

Algorithm 1: BIASEDSAMPLE

Input: Original tensor $\underline{\mathbf{X}}$ of size $I \times J \times K$, sampling factor s.
Output: Sampled tensor $\underline{\mathbf{X}}_s$, index sets $\mathcal{I}, \mathcal{J}, \mathcal{N}$.

1: Compute
$$\mathbf{x}_a(i) = \sum_{j=1}^{J}\sum_{k=1}^{K}\underline{\mathbf{X}}(i,j,k),\ \mathbf{x}_b(j) = \sum_{i=1}^{I}\sum_{k=1}^{K}\underline{\mathbf{X}}(i,j,k),\ \mathbf{x}_c(k) = \sum_{i=1}^{I}\sum_{j=1}^{J}\underline{\mathbf{X}}(i,j,k).$$
2: Compute set of indices $\mathcal{I}$ as random sample without replacement of $\{1 \cdots I\}$ of size I/s with probability $p_{\mathcal{I}}(i) = \mathbf{x}_a(i) / \sum_{i=1}^{I} \mathbf{x}_a(i)$. Likewise for $\mathcal{J}, \mathcal{K}$.
3: Return $\underline{\mathbf{X}}_s = \underline{\mathbf{X}}(\mathcal{I}, \mathcal{J}, \mathcal{K})$.

by-product of our approach is *sparsity on the factors by construction*, thereby satisfying Goal **G2**.

However, Algorithm 2 relies on a sole sample of the tensor and it might be the case that some significant portions of the data, depending on the sampling factor and the data distribution, may be left out. To that end, we introduce Algorithm 3 which is our main contribution. Algorithm 3 generates many samples and correctly combines them, in order to achieve better extraction of the true latent factors of the data tensor.

The key idea behind Algorithm 3 is the method by which all the different samples are combined in order to output the decomposition matrices; more specifically, intuitively we enforce all the different samples to have a common set of indices $\mathcal{I}_p, \mathcal{J}_p, \mathcal{K}_p$ (which is a p fraction of the whole sample). Having this common basis, we are able to combine the samples using Algorithm 4. The basic idea of Algorithm 4 is the following: We arbitrarily choose the factors of one sample to serve as reference, and we distribute their coefficients to the corresponding indices of the factor matrices of the original tensor, as in Algorithm 2. We then process each of the remaining samples individually. For each one of them, we establish a correspondence of the sampled factors to the reference factors, and we update the zero coefficients of the reference factors using values from the current sample. A fairly subtle issue that arises is how to overcome scaling disparities between factors coming from two different samples. Key here, as described in line 5 of Algorithm 3, is to counter-scale the two merge candidates, using only the norms of the common parts indexed by $\mathcal{I}_p, \mathcal{J}_p, \mathcal{K}_p$; by doing so, the common parts will be scaled to unit norm, and the rest of the vectors will also refer to the correct, same scaling, thereby effectively resolving scaling correspondence.

Note that the generation of the r distinct samples of $\underline{\mathbf{X}}$, as well as the PARAFAC decomposition of each of them may be carried out in parallel; thus satisfying Goal **G1**. Regarding Goal **G3**, note that correctness of the merge operation requires certain conditions; it cannot be guaranteed when the individual random samples do not satisfy PARAFAC identifiability conditions, or when the common piece that is used as a reference for merging is too small (p is too low). Proposition 1 provides a first correctness result for our merging algorithm.

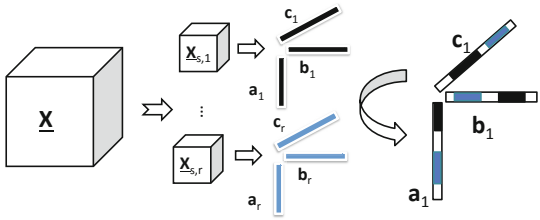


Fig. 3. Example of rank-1 PARAFAC using PARCUBE (Algorithm 3). The procedure described is the following: Create r independent samples of $\underline{\mathbf{X}}$, using Algorithm 1. Run the PARAFAC- ALS algorithm for $K = 1$ and obtain r triplets of vectors, corresponding to the first component of $\underline{\mathbf{X}}$. As a final step, combine those r triplets, by distributing their values to the original sized triplets, as indicated in Algorithm 3.

Algorithm 2: Basic PARCUBE for Non-negative PARAFAC

Input: Tensor $\underline{\mathbf{X}}$ of size $I \times J \times K$, number of components F, sampling factor s.
Output: Factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of size $I \times F$, $J \times F$, $K \times F$ respectively.
1: Run BIASEDSAMPLE ($\underline{\mathbf{X}}, s$) (Algorithm 1) and obtain $\underline{\mathbf{X}}_s$ and $\mathcal{I}, \mathcal{J}, \mathcal{K}$.
2: Run Non-Negative PARAFAC ($\underline{\mathbf{X}}_s, F$) and obtain $\mathbf{A}_s, \mathbf{B}_s, \mathbf{C}_s$ of size $I/s \times F$, $J/s \times F$ and $K/s \times F$.
3: $\mathbf{A}(\mathcal{I}, :) = \mathbf{A}_s$, $\mathbf{B}(\mathcal{J}, :) = \mathbf{B}_s$, $\mathbf{C}(\mathcal{K}, :) = \mathbf{C}_s$

Algorithm 3: PARCUBE for Non-negative PARAFAC with repetition

Input: Tensor $\underline{\mathbf{X}}$ of size $I \times J \times K$, number of components F, sampling factor s, number of repetitions r.
Output: PARAFAC factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of size $I \times F$, $J \times F$, $K \times F$ respectively and vector $\boldsymbol{\lambda}$ of size $F \times 1$ which contains the scale of each component.
1: Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$ to all-zeros.
2: Randomly, *using mode densities as bias*, select a set of $100p\%$ ($p \in [0, 1]$) indices $\mathcal{I}_p, \mathcal{J}_p, \mathcal{K}_p$ to be common across all repetitions.
3: **for** $i = 1 \cdots r$ **do**
4: Run Algorithm 2 with sampling factor s, using $\mathcal{I}_p, \mathcal{J}_p, \mathcal{K}_p$ as a common reference among all r different samples and obtain $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$. The sampling is made on the set difference of the set of all indices and the set of common indices.
5: Calculate the ℓ_2 norm of the columns of the common part: $\mathbf{n}_a(f) = \|\mathbf{A}_i(\mathcal{I}_p, f)\|_2$, $\mathbf{n}_b(f) = \|\mathbf{B}_i(\mathcal{J}_p, f)\|_2$, $\mathbf{n}_c(f) = \|\mathbf{C}_i(\mathcal{K}_p, f)\|_2$ for $f = 1 \cdots F$. Normalize columns of $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$ using $\mathbf{n}_a, \mathbf{n}_b, \mathbf{n}_c$ and set $\boldsymbol{\lambda}_i(f) = \mathbf{n}_a(f)\mathbf{n}_b(f)\mathbf{n}_c(f)$. Note that the common part will now be normalized to unit norm.
6: **end for**
7: $\mathbf{A}$ =FACTORMERGE ($\mathbf{A}_i$), $\mathbf{B}$ =FACTORMERGE ($\mathbf{B}_i$),$\mathbf{C}$ =FACTORMERGE ($\mathbf{C}_i$)
8: $\boldsymbol{\lambda}$ = average of $\boldsymbol{\lambda}_i$.

Algorithm 4: FACTORMERGE

Input: Factor matrices $\mathbf{A}_i$ of size $I \times F$ each, where $i = 1 \cdots r$, and r is the number of repetitions, $\mathcal{I}_p$: set of common indices.
Output: Factor matrix $\mathbf{A}$ of size $I \times F$.

```
Set A = A_1
for i = 2 ··· r do
    for f_1 = 1 ··· F do
        for f_2 = 1 ··· F do
            Compute similarity v(f_2) = (A(I_p, f_2))^T (A_i(I_p, f_1)))
        end for
        c = arg max_{c'} v(c')
        Update only the zero entries of A(:, c) using vector A_i(:, f_1).
    end for
end for
```

Proposition 1. *Let* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *be the* PARAFAC *decomposition of* $\underline{\mathbf{X}}$*, and assume that* $\mathbf{A}(\mathcal{I}_p, :)$ *(*$\mathbf{A}$ *restricted to the common* I*-mode reference rows) is such that any two of its columns are linearly independent; and likewise for* $\mathbf{B}(\mathcal{J}_p, :)$ *and* $\mathbf{C}(\mathcal{K}_p, :)$*. Note that if* $\mathbf{A}(\mathcal{I}_p, :)$ *has as few as 2 rows (*$|\mathcal{I}_p| \geq 2$*) and is drawn from a jointly continuous distribution, this requirement on* $\mathbf{A}(\mathcal{I}_p, :)$ *is satisfied with probability 1. Further assume that each of the sub-sampled models is identifiable, and the true underlying rank-one (punctured) factors are recovered, up to permutation and scaling, from each sub-sampled dataset. Then Algorithm 4 is able to merge the factors coming from the different samples of the tensor correctly, i.e., is able to find the correct correspondence between the columns of the factor matrices* $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$*.*

Proof sketch 1 *Consider the common part of the* $\mathbf{A}$*-mode loadings recovered from the different sub-sampled versions of* $\underline{\mathbf{X}}$*: under the foregoing assumptions, the* $\mathbf{A}_i(\mathcal{I}_p, :)$ *will be permuted and column-scaled versions of* $\mathbf{A}(\mathcal{I}_p, :)$*. After scaling* the common part *of each column to unit norm, Algorithm 4 seeks to match the permutations by maximizing correlation between pairs of columns drawn from* $\mathbf{A}_i(\mathcal{I}_p, :)$ *and* $\mathbf{A}_j(\mathcal{I}_p, :)$*. From the Cauchy-Schwartz inequality, correlation between any two unit-norm columns is* ≤ 1*, and equality is achieved only when the correct columns are matched, because any two distinct columns of the underlying* $\mathbf{A}(\mathcal{I}_p, :)$ *are linearly independent. Furthermore, by normalizing the scales of the matched columns to equalize the norm of the common reference part, the insertions that follow include the correct scaling too. This shows that Algorithm 4 works correctly in this case.* ■

The above proposition serves as a sanity check for correctness. In reality, there will be noise and other imperfections that come into play, implying that the punctured factor estimates will at best be approximate. This implies that a larger common sample size ($|\mathcal{I}_p| \geq 2$, $|\mathcal{J}_p| \geq 2$, $|\mathcal{K}_p| \geq 2$) will generally help Algorithm 4 to correctly merge the pieces coming from the different samples. We have carried out extensive experiments verifying that Algorithm 4 works

well in practice, under common imperfections. Those experiments also suggest that increasing the number of samples, r, reduces the PARAFAC approximation error.

4 Experiments and Discoveries

In this section we provide experimental evaluation of our proposed method. First, we evaluate the performance of PARCUBE, compared to the current state of the art for handling sparse tensors in Matlab, i.e. the Tensor Toolbox for Matlab [8]. Since our algorithm, by construction, tends to output sparse factors, we also evaluate the validity of that claim by comparing the degree of sparsity of the output to the one given by the Tensor Toolbox and the one given by PARAFAC SLF [19], which is the state of the art for PARAFAC decompositions with sparsity on the latent factors. The speedups we are reporting were measured on a 2.7 GHz Intel Core i5 with 4GB of RAM. Finally, we apply our approach in order to analyze real datasets presented in Table 1.

Table 1. Datasets analyzed

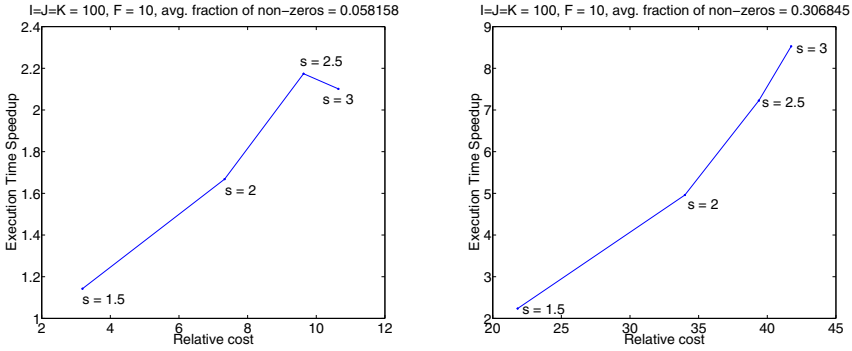
Name	Description	Dimensions	NNZ
ENRON [1]	(sender, recipient, month)	$186 \times 186 \times 44$	9838
LBNL [18]	(src, dst, port #)	$65170 \times 65170 \times 65327$	27269
FACEBOOK [25]	(wall owner, poster, day)	$63891 \times 63890 \times 1847$	737778
NELL [2]	(noun-phrase, noun-phrase, context)	$14545 \times 14545 \times 28818$	76879419

We implemented PARCUBE in Matlab, and we make it available through the first author's web site [2]. We furthermore use the Tensor Toolbox for Matlab [8] as our core PARAFAC decomposition implementation.

4.1 Performance and Scalability Evaluation

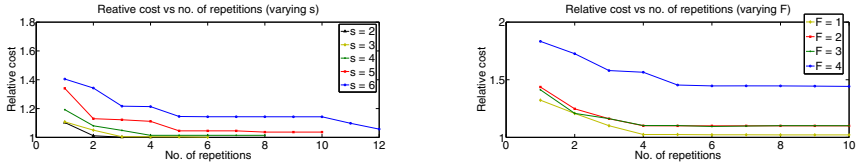
In the following lines, we evaluate the performance of PARAFAC using PARCUBE (Algorithm 2). As a performance metric, we use the the relative cost of the PARAFAC model, i.e. the cost of the model using our sampling approach, divided by the cost of fitting a PARAFAC model using the original tensor. In Fig. 4, we measure the relative cost as a function of the speedup incurred by using our PARCUBE, for different values of the sampling factor; this experiment was carried out on $100 \times 100 \times 100$ randomly generated, synthetic tensors, as we required full control over the true number of components and the degree of sparsity for each component. In Fig. 5, we show the relative cost using the ENRON dataset, for various numbers of repetitions (i.e. distinct samples). We see, in this case, that as the number of repetitions increases, the approximation improves, as expected, from our theoretical result.

[2] Download PARCUBE at `www.cs.cmu.edu/~epapalex/src/parCube.zip`



(a) (b)

Fig. 4. ParCube is faster than ALS-Parafac: Speedup vs Relative cost (ParCube/ ALS-Parafac) for 1 repetition, for varying sampling factor and different degrees of sparsity. We observe that even for a relatively high sampling factor, we get relatively good relative cost, which may be further improved using repetition. Key here is that by using repetition, because this procedure may be carried out in parallel, we may improve the accuracy and maintain similar speedup.



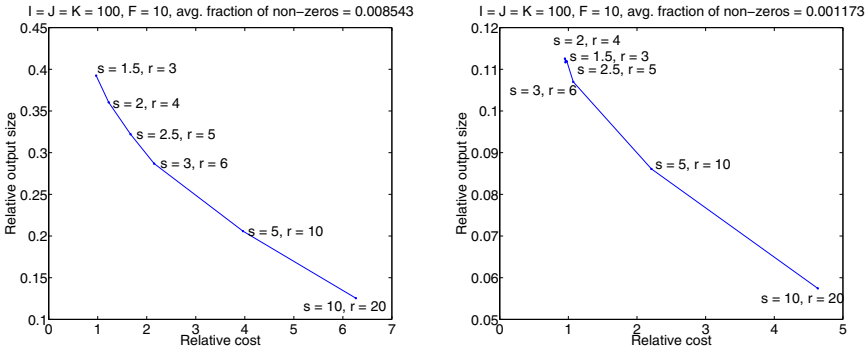
(a) Enron: Relative cost vs No. of repetitons (varying s)

(b) Enron: Relative cost vs No. of repetitons (varying F)

Fig. 5. ParCube reduces the Parafac approximation cost: (a) Approximation cost vs number of repetitions for varying s, where $r = 2s$ (b) Approximation cost vs number of repetitions for varying F and fixed $s = 5$. In both cases, the approximation improves as r increases, as expected

4.2 Factor Sparsity Assessment

In Fig. 6, we measure the relative output size (i.e. the relative degree of sparsity) between ParCube and Tensor Toolbox non-negative Parafac. The output size is simply defined as $NNZ(\mathbf{A}) + NNZ(\mathbf{B}) + NNZ(\mathbf{C})$, which clearly reflects the degree of sparsity in the decomposition factors. In Fig. 7 we measure the relative output size between ParCube and Parafac SLF, as a function of the sampling factor s, for different values of the sparsifying parameter λ used by Parafac SLF (more details in [19]).



(a) (b)

Fig. 6. ParCube outputs sparse factors: Relative Output size (ParCube/ ALS-Parafac) vs Relative cost. We see that the results of ParCube are more than 90% sparser than the ones from Tensor Toolbox, while maintaining the same approximation error.

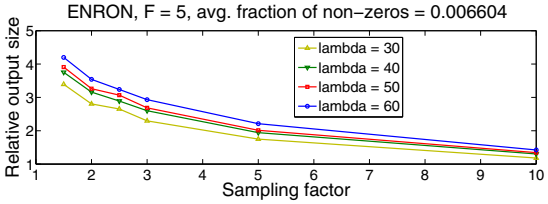


Fig. 7. ParCube outputs sparse factors: Relative Output size (ParCube/ Parafac SLF) vs sampling factor s (where no. of repetitions is $r = 2s$.

4.3 Parallelizability

As we have mentioned above, lines 3 to 6 of Algorithm 3 may be carried out entirely in parallel; we have also established, in the previous subsection, that by doing more repetitions, we largely improve the approximation error, and in particular, we converge to the approximation error of the ALS-Parafac algorithm. In its current version, ParCube is not implemented to run in multiple machines, however, here we discuss the potential merits of such an implementation. For evaluation purposes, we divided the run time of Algorithm 3 to a parallelizable (lines 3 to 6) and a serial part (everything else). For r repetitions, we added the serial run-time with the *maximum* of the r different, parallel running times. This way, we are loosely emulating the run-time that we would get if we used r cores or machines to run the Algorithm. In particular, by conducting experiments on $256 \times 256 \times 256$ tensors with 0.0578 fraction of non-zeros on average, we got 1.2 average speedup with relative error 1.67 (for $s = 2, r = 4$) and 14.2 speedup with relative error 5.9 (for $s = 10, r = 20$).

4.4 ParCube at Work

In this section we present interesting patterns and anomalies, that we were able to discover in the datasets of Table 1, demonstrating that our proposed algorithm ParCube is both practical and effective for data mining practitioners. So far, we don't have an automated method for the selection of parameters s, r, and p, but we leave this for future work; the choice is now made empirically.

ENRON. This very well known dataset contains records for 44 months (between 1998 and 2002) of the number of emails exchanged between the 184 employees of the company, forming a $184 \times 184 \times 44$ of 9838 non-zero entries. We executed Algorithm 3 using $s = 2$ and $r = 4$ and we applied similar analysis to the resulting factors as the one applied in [7,19]. In Figure 8 we illustrate the temporal evolution of the 4 most prevailing groups in our analysis, having annotated the figure with important events, corresponding to peaks in the communication activity. Labelling of the groups was done manually; because the factors were not very sparse we filtered out very low values on each factor. This issue most certainly stems from the fact that this dataset is not particularly large and therefore by applying the regular ALS-Parafac algorithm to the samples (which is known to yield dense factors), we end up with dense sample factors, which eventually, due to repetition, tend to cover most of the data points. This, however, was not the case for larger datasets analyzed in the following lines, for which the factors turned out to be extremely sparse.

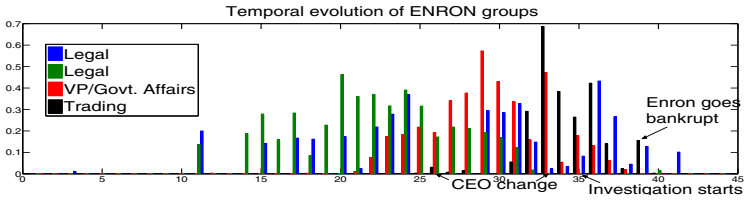


Fig. 8. Temporal evolution of 4 groups in the Enron dataset. We have labelled the groups, according to the position of the participants in the company. The labels of the extracted groups are consistent with other works in the literature [7,19], albeit they have been extracted with somewhat different order. We have also discovered 2 'Legal' groups that behave slightly differently over time, a fact probably stemming from the different people involved in each group.

LBNL Network Traffic. This dataset consists of (source, destination, port #) triplets, where each value of the corresponding tensor is the number of packets sent. The snapshot of the dataset we used, formed a $65170 \times 65170 \times 65327$ tensor of 27269 non-zeros. We ran Algorithm 3 using $s = 5$ and $r = 10$ and we were able to identify what appears to be a port-scanning attack: The component shown in Fig. 9 contains only one source address (addr. 29571), contacting one destination address (addr. 30483) using a wide range of near-consecutive ports

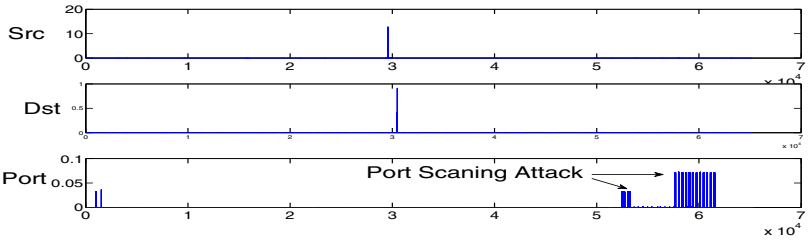
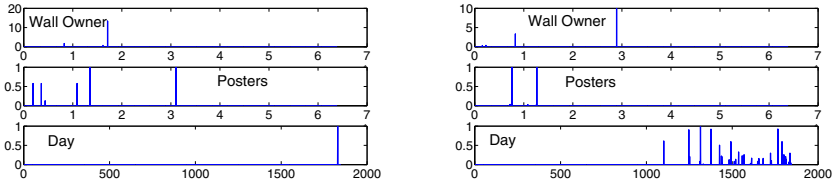


Fig. 9. Anomaly on the LBNL data: We have one source address (addr. 29571), contacting one destination address (addr. 30483) using a wide range of near-consecutive ports, possibly indicating a port scanning attack

(while sending the same amount of packets to each port), a behaviour which should certainly raise a flag to the network administrator, indicating a possible port-scanning attack.

Facebook Wall Posts. This dataset [3] first appeared in [25]; the specific part of the dataset we used consists of triplets of the form (Wall owner, Poster, day), where the Poster created a post on the Wall owner's Wall on the specified timestamp. By choosing daily granularity, we formed a $63891 \times 63890 \times 1847$ tensor, comprised of 737778 non-zero entries; subsequently, we ran Algorithm 3 using $s = 100$ and $r = 10$. In Figure 10 we present our most surprising findings: On the left subfigure, we demonstrate what appears to be the Wall owner's birthday, since many posters posted on a single day on this person's Wall; this event may well be characterized as an "anomaly". On the right subfigure, we demonstrate what "normal" FACEBOOK activity looks like.



(a) FACEBOOK anomaly (Wall owner's birthday)

(b) FACEBOOK normal activity

Fig. 10. Results for FACEBOOK using $s = 100, r = 10, F = 15$. Subfigure (a): FACEBOOK "anomaly": One Wall, many posters and only one day. This possibly indicates the birthday of the Wall owner. Subfigure(b): FACEBOOK "normal" activity: Many users post on many users' Walls, having a continuous daily activity.

[3] Download FACEBOOK at http://socialnetworks.mpi-sws.org/data-wosn2009.html

NELL. This dataset consists of triplets of the form (noun-phrase, noun-phrase, context). which form a tensor with assorted modes of size $14545 \times 14545 \times 28818$ and 76879419 non-zeros, and as values the number of occurrences of each triplet. The context phrase may be just a verb or a whole sentence. After computing the PARAFAC decomposition of the tensor using PARCUBE with $s = 500$, and $r = 10$ repetitions, we computed the noun-phrase similarity matrix $\mathbf{A}\mathbf{A}^T + \mathbf{B}\mathbf{B}^T$ and out of that, we were able to discover potential synonyms to noun-phrases, that we report on Table 2.

Table 2. NELL: Potential synonym discovery

Noun-phrase	Potential Synonyms
computer	development
period	day, life
months	life
facilities	families, people, communities
rooms	facilities
legs	people
communities	facilities, families, students

5 Related Work

Tensor Applications. Tensors and tensor decompositions have gained increasing popularity in the last few years, in the data mining community [14]. The list of tensor applications in data mining is long, however we single out a few that we deemed representative: In [13], the authors extend the well known link analysis algorithm HITS, incorporating textual/topical information. In [7] and [6] the authors use tensors for social network analysis on the ENRON dataset. In [22], the authors propose a sampling-based Tucker3 decomposition in order to perform content based network analysis and visualization. The list continues, including applications such as [10] [16] [3]. Apart from Data Mining, tensors have been and are still being applied in a multitude of fields such as Chemometrics [9] and Signal Processing [21].

State of the Art Toolboxes. The standard framework for working with tensors is Matlab; there exist two toolboxes, both of very high quality: The Tensor Toolbox for Matlab[5,8] (specializing in sparse tensors) and the N-Way Toolbox for Matlab [4] (specializing in dense tensors).

Relevant Approaches. In [20], the authors propose a partition-and-merge scheme for the PARAFAC decomposition which, however, does not offer factor sparsity. In [19], the authors introduce a PARAFAC decomposition with latent factor sparsity. In [17] and [23] we find two interesting approaches, where a tensor is viewed as a stream and the challenge is to track the decomposition. In terms of parallel algorithms, [26] introduces a parallel Non-negative Tensor Factorization. Finally, [24,22] propose randomized, sampling based Tucker3 decompositions.

6 Conclusion

In this work we have introduced ParCube, a new, fast, parallelizable tensor decomposition which produces sparse factors by construction. Furthermore, it enables processing of large tensors that may not fit in memory. We provide theoretical results that indicate correctness of our algorithm; one of our core contributions pertains to the correct merging of the individual samples. We have demonstrated its merits with respect to sparsity and speedup, compared to the current state of the art, through extensive experimentation. Moreover, the speedup benefits of ParCube may be further improved if we exploit its massive parallelizability. Finally, the practicality of ParCube is heavily pronounced by analyzing four different real datasets, discovering patterns and anomalies.

References

1. Enron e-mail dataset, http://www.cs.cmu.edu/~enron/
2. Read the web, http://rtw.ml.cmu.edu/rtw/
3. Acar, E., Aykut-Bingol, C., Bingol, H., Bro, R., Yener, B.: Multiway analysis of epilepsy tensors. Bioinformatics 23(13), i10–i18 (2007)
4. Andersson, C.A., Bro, R.: The n-way toolbox for matlab. Chemometrics and Intelligent Laboratory Systems 52(1), 1–4 (2000)
5. Bader, B.W., Kolda, T.G.: Efficient MATLAB computations with sparse and factored tensors. SIAM Journal on Scientific Computing 30(1), 205–231 (2007)
6. Bader, B.W., Berry, M.W., Browne, M.: Discussion tracking in enron email using parafac. Survey of Text Mining II, 147–163 (2008)
7. Bader, B.W., Harshman, R.A., Kolda, T.G.: Temporal analysis of social networks using three-way dedicom. Sandia National Laboratories TR SAND2006-2161 (2006)
8. Bader, B.W., Kolda, T.G.: Matlab tensor toolbox version 2.2. Sandia National Laboratories, Albuquerque (2007)
9. Bro, R.: Parafac. tutorial and applications. Chemometrics and Intelligent Laboratory Systems 38(2), 149–171 (1997)
10. Chew, P.A., Bader, B.W., Kolda, T.G., Abdelali, A.: Cross-language information retrieval using parafac2. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 143–152. ACM (2007)
11. Drineas, P., Kannan, R., Mahoney, M.W., et al.: Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. SIAM Journal on Computing 36(1), 184 (2006)
12. Harshman, R.A.: Foundations of the parafac procedure: Models and conditions for an "explanatory" multimodal factor analysis (1970)
13. Kolda, T.G., Bader, B.W.: The tophits model for higher-order web link analysis. In: Workshop on Link Analysis, Counterterrorism and Security, vol. 7, pp. 26–29 (2006)
14. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review 51(3) (2009)
15. Mahoney, M.W., Maggioni, M., Drineas, P.: Tensor-cur decompositions for tensor-based data. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 327–336. ACM (2006)

16. Maruhashi, K., Guo, F., Faloutsos, C.: Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In: Proceedings of the Third International Conference on Advances in Social Network Analysis and Mining (2011)
17. Nion, D., Sidiropoulos, N.D.: Adaptive algorithms to track the parafac decomposition of a third-order tensor. IEEE Transactions on Signal Processing 57(6), 2299–2310 (2009)
18. Pang, R., Allman, M., Bennett, M., Lee, J., Paxson, V., Tierney, B.: A first look at modern enterprise traffic. In: Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, p. 2. USENIX Association (2005)
19. Papalexakis, E.E., Sidiropoulos, N.D.: Co-clustering as multilinear decomposition with sparse latent factors. In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2064–2067. IEEE (2011)
20. Phan, A.H., Cichocki, A.: Block decomposition for very large-scale nonnegative tensor factorization. In: 2009 3rd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), pp. 316–319. IEEE (2009)
21. Sidiropoulos, N.D., Giannakis, G.B., Bro, R.: Blind parafac receivers for ds-cdma systems. IEEE Transactions on Signal Processing 48(3), 810–823 (2000)
22. Sun, J., Papadimitriou, S., Lin, C.Y., Cao, N., Liu, S., Qian, W.: Multivis: Content-based social network exploration through multi-way visual analysis. In: Proc. SDM, vol. 9, pp. 1063–1074 (2009)
23. Sun, J., Tao, D., Faloutsos, C.: Beyond streams and graphs: dynamic tensor analysis. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 374–383. ACM (2006)
24. Tsourakakis, C.E.: Mach: Fast randomized tensor decompositions, Arxiv preprint arXiv:0909.4969 (2009)
25. Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN 2009) (August 2009)
26. Zhang, Q., Berry, M.W., Lamb, B.T., Samuel, T.: A Parallel Nonnegative Tensor Factorization Algorithm for Mining Global Climate Data. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009, Part I. LNCS, vol. 5545, pp. 405–415. Springer, Heidelberg (2009)

Stochastic Coordinate Descent Methods for Regularized Smooth and Nonsmooth Losses

Qing Tao[1,2], Kang Kong[1], Dejun Chu[1], and Gaowei Wu[2]

[1] New Star Research Inst. of Applied Technology, Hefei 230031, P.R. China
ln.kang.kong,fangboc@gmail.com
[2] Inst. of Automation, Chinese Academy of Sciences, Beijing, 1000190, P.R. China
{qing.tao,gaowei.wu}@ia.ac.cn

Abstract. Stochastic Coordinate Descent (SCD) methods are among the first optimization schemes suggested for efficiently solving large scale problems. However, until now, there exists a gap between the convergence rate analysis and practical SCD algorithms for general smooth losses and there is no primal SCD algorithm for nonsmooth losses. In this paper, we discuss these issues using the recently developed structural optimization techniques. In particular, we first present a principled and practical SCD algorithm for regularized smooth losses, in which the one-variable subproblem is solved using the proximal gradient method and the adaptive componentwise Lipschitz constant is obtained employing the line search strategy. When the loss is nonsmooth, we present a novel SCD algorithm, in which the one-variable subproblem is solved using the dual averaging method. We show that our algorithms exploit the regularization structure and achieve several optimal convergence rates that are standard in the literature. The experiments demonstrate the expected efficiency of our SCD algorithms in both smooth and nonsmooth cases.

Keywords: Optimization Algorithms, Coordinate Descent Algorithms, Nonsmooth and smooth Losses, Large-Scale Learning.

1 Introduction

Given a training set $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, where $(\mathbf{x}_i, y_i) \in \mathbb{R}^N \times Y, Y = \{-1, 1\}$, $\mathbf{x}_i$ is independently drawn and identically distributed, and y_i is the label of $\mathbf{x}_i$. The task of regularized learning is usually cast as the following convex optimization problem,

$$F(\mathbf{w}) = \lambda P(\mathbf{w}) + \sum_{i=1}^{m} f_i(\mathbf{w}) \tag{1}$$

where λ is a trade-off parameter, $P(\mathbf{w})$ is a simple regularizer (such as l_1 or l_2 norm) and $f_i(\mathbf{w})$ is the loss caused by $(\mathbf{x}_i, y_i)$. If the gradient of each $f_i(\mathbf{w})$ is Lipschitz continuous, we call (1) a regularized smooth problem. In the literature [3,6,26], $f_i(\mathbf{w}) = \max\{0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i\rangle\}^2$ is usually referred to as L_2-loss and $f_i(\mathbf{w}) = \max\{0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i\rangle\}$ is L_1-loss.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 537–552, 2012.

Using Coordinate Descent (CD) methods to solve optimization problems has a long history and we refer readers to [24] and [26] that summarize previous work and present comparison for various kinds of CD algorithms. In machine learning, the primal CD operates by sequentially drawing all features, one at a time, and adjusting the learning variables using the closed-form solvers that are based on the single feature only. More precisely, the process from $\mathbf{w}^t$ to $\mathbf{w}^{t+1}$ is called an outer iteration. In each outer iteration, there are N inner iterations so that sequentially the component $w_1^t, w_2^t, \cdots, w_N^t$ are updated. As indicated in [18,19], the low computational complexity at per iteration, the inherently fresh information of updated features and cheap computation of coordinate directional derivatives make CD one of the most efficient optimization techniques in dealing with sparse huge scale problems. In the huge-scale problems in the sense that even the problem's data may be only partially available at the moment of evaluating the current test point, going over all dimensions causes an expensive outer iteration. Instead, one can randomly update only one component of $\mathbf{w}$ at each outer iteration. This kind of methods is referred to as Stochastic CD (SCD).

The practical efficiency of CD has been shown by extensive comparison experiments and an important fact is that the dual CD method for linear SVM in [6] performs very well on large scale document data [26]. However, as pointed out in [3], one should use primal CD when the number of features is much smaller than the number of instances. Although the primal CD methods for regularized learning [3,19,20,24,26] has been receiving much attention, some problems still exist. First, for a general smooth loss especially the commonly used L_2-loss, the existing results either just prove convergence rates [20] or only provide practical algorithms [26] due to the lack of the strong convexity. How to fill the gap between the convergence rate analysis and practical efficiency is still an emergent question. Second, for a nonsmooth loss such as the popularly used hinge loss, there is still no primal SCD algorithm due to the lack of the differentiability. In this paper, we discuss these issues using the structural optimization techniques.

Recently, many remarkable achievements have been made in the area of structural convex optimization [16]. It has been shown that the black-box gradient-type optimization approaches can be replaced by optimization techniques based on a clever use of problem's structure. For nonsmooth problems, Nesterov proposed a primal-dual subgradient method [17] to take the place of the classical Projected Subgradient Algorithm (PSA, [1]). In all situations, this method was proved to be optimal from the viewpoint of worst-case black-box lower complexity bounds. Further, if the objective function is smooth, Nesterov presented novel smooth convex optimization algorithms whose rate of convergence achieve the optimal $O(1/t^2)$ convergence rate in a seminal work [13]. Other variants of this method for minimizing composite objective functions of the form are called APG in [23] and FISTA in [2]. By extending the dual averaging scheme in [17] to regularized problems, an online Regularized Dual Averaging (RDA) method for solving regularized problem (1) was obtained in [25]. In the case of l_1 regularization, RDA can particularly exploit the regularization structure and effectively

obtain the sparse solutions. More recently, by using PG, Nesterov proposed an efficient SCD scheme for solving huge-scale smooth optimization problems [18].

In this paper, we present a unified framework for developing CD algorithms from the smooth and nonsmooth structural optimization methods. In particular, for smooth losses, we first extend the smooth CD algorithm in [18] to solve regularized smooth problems with nonsmooth regularizers by using PG. Then we consider the important question of decreasing the factor in the convergence rate and derive an adaptive SCD algorithm using the line search technique. Second, for nonsmooth losses, we present a novel SCD algorithm for regularized nonsmooth losses, in which the randomly selected one-variable subproblem is solved using the RDA method. Since we separately treat the regularizer and loss, all our SCD algorithms can effectively exploit the regularization structure. Theoretical analysis shows that we achieve several optimal rates which are standard in the literature especially for convex and strongly convex problems. Our nonsmooth SCD expands the field of SCD and our smooth SCD fills the gap between convergent rate analysis [20] and practical algorithms [26] especially for L_2-loss.

The experiments show that our nonsmooth SCD outperforms the state-of-the-art solvers in [17,4,21,6]. For regularized smooth loss problems, the toy experiments illustrate that our adaptive smooth SCD outperforms the state-of-the-art solver in [2] while the real experiments demonstrate that it has the same practicality as the state-of-the-art solver in [26].

The rest of this paper is organized as follows. Section 2 and Section 3 discuss SCD algorithms for smooth and nonsmooth losses respectively. Experimental results are reported in Section 4 and conclusions are made in the last section.

2 SCD Algorithms for Smooth Losses

It is well-known that the complexity of a convex optimization problem is closely linked with its level of smoothness on the first derivatives of functional components [16]. If more information on the smoothness about f is available, higher performance algorithms are expected. In this section, we assume that ∇f is Lipschitz continuous. The Lipschitz condition of ∇f measures how well f is approximated at some point by its linearization.

For a primal CD method, the optimization process starts from an initial point $\mathbf{w}^0$ and generates a sequence of vectors $\{\mathbf{w}^t\}$. Each outer iteration generates vectors $\mathbf{w}^{t,i} \in \mathbb{R}^N$, $i = 1, 2, \cdots, N+1$, such that $\mathbf{w}^{t,1} = \mathbf{w}^t$, $\mathbf{w}^{t,N+1} = \mathbf{w}^{t+1}$ and $\mathbf{w}^{t,i} = [w_1^{t+1}, \cdots, w_{i-1}^{t+1}, w_i^t, \cdots, w_N^t]^T, \forall i = 2, \cdots, N$. The concerned one-variable sub-problem is $\min_z \lambda P(\mathbf{w}^{t,i} + z\mathbf{e}_i) + a_i(z)$, where $a_i(z) = f(\mathbf{w}^{t,i} + z\mathbf{e}_i)$ and $\mathbf{e}_i = [0, \cdots, 0, 1, 0, \cdots, 0]^T$.

As L_2-loss is only Lipschitz differentiable but not twice differentiable and $P(\mathbf{w}) = \|\mathbf{w}\|_1$ is non-differentiable, certain special considerations in generalizing the second derivative are given in [3]. First, in order to derive the closed-form solution of single-variable sub-problem for L_2-loss, the following second-order approximation of the loss term is adopted

$$\min_z a_i'(0)z + \frac{1}{2}a_i''(0)z^2 + \lambda p(w_i^{t,i} + z) - \lambda p(w_i^{t,i}) \tag{2}$$

where $a_i''(0)$ is the generalized second derivative defined in [26] and [3], $p(\omega) = \omega^2$ when $P(\mathbf{w}) = \|\mathbf{w}\|_2^2$ and $p(\omega) = |\omega|$ when $P(\mathbf{w}) = \|\mathbf{w}\|_1$. On the other hand, to ensure the convergence of l_1 regularized CD algorithms, the line search strategy in [24] is modified in [26] to find γ,

$$F(\mathbf{w}^{t,i} + \gamma d\mathbf{e}_i) - F(\mathbf{w}^{t,i}) \leq \sigma\gamma[a_i'(0)d + \lambda|w_i^{t,i} + d| - \lambda|w_i^{t,i}|] \tag{3}$$

where d is the solution of (2), σ is any constant in $(0, 1)$, $\beta \in (0, 1)$, and $\gamma = \max\{1, \beta, \beta^2, \ldots, \}$ such that γd satisfies (3). By solving (2) and (3), the inner update from $\mathbf{w}^{t,i}$ to $\mathbf{w}^{t,i+1}$ is $w_i^{t,i+1} = w_i^{t,i} + \gamma d$.

In the following, we will construct CD based on the gradient methods for smooth functions. Obviously, there exist a constant $L(f) > 0$ such that $\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{u})\| \leq L(f)\|\mathbf{w} - \mathbf{u}\|, \forall \mathbf{w}, \mathbf{u} \in \mathbb{R}^N$. Then, for any $L \geq L(f)$,

$$f(\mathbf{w}) \leq f(\mathbf{u}) + \langle \mathbf{w} - \mathbf{u}, \nabla f(\mathbf{u})\rangle + (L/2)\|\mathbf{w} - \mathbf{u}\|^2 \tag{4}$$

for every $\mathbf{w}, \mathbf{u} \in \mathbb{R}^N$. Let $l_f(\mathbf{w}, \mathbf{u}) = f(\mathbf{u}) + \langle \mathbf{w} - \mathbf{u}, \nabla f(\mathbf{u})\rangle$. The key operation in PG [23] is

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} \{l_f(\mathbf{w}, \mathbf{w}^t) + \lambda P(\mathbf{w}) + (L/2)\|\mathbf{w} - \mathbf{w}^t\|^2\} \tag{5}$$

It is easy to find that (5) is separable and has an entry-wise closed-form solution. Motivated by the CD for L_2-loss in [3], we can solve (5) in only one of the N components at each step. To be more precise, the randomly selected one-variable sub-problem now is

$$w_i^{t+1} = \arg\min_{\omega} \left\{\omega(g_i^t - Lw_i^t) + \lambda p(\omega) + (L/2)\omega^2\right\} \tag{6}$$

If d is the solution of (6), the update from $\mathbf{w}^t$ to $\mathbf{w}^{t+1}$ is $\mathbf{w}^{t+1} = \mathbf{w}^t + d\mathbf{e}_i$. We describe our SCD algorithm for regularized smooth losses in Algorithm 1.

Algorithm 1. Smooth SCD

Initialize a weight vector $\mathbf{w}^1$.
repeat
1. Choose $i \in \{1, 2, \ldots, N\}$ uniformly at random
2. calculate g_i^t
3. solve (6) and update $\mathbf{w}^{t+1}$
4. $t := t + 1$

until a stopping condition is satisfied

If we let $\lambda = 0$ or $P(\mathbf{w})$ be the indicator function of a closed convex set, Smooth SCD recovers the SCD scheme including both unconstrained and constrained minimizations in [18]. After t iterations, Nonsmooth SCD generates a random output $[\mathbf{w}^t, F(\mathbf{w}^t)]$. Obviously, $[\mathbf{w}^t, F(\mathbf{w}^t)]$ depends on the random variable $\xi_t = \{i_0, i_1, i_2, \ldots i_t\}$, where i_t is independently and randomly chosen from

the set $\{1, 2, \cdots, N\}$ with probability $1/N$. We denote $\phi_t = \mathbf{E}_{\xi_{t-1}} F(\mathbf{w}^t)$. In the following, we extend the convergence theorem in [18] to regularized learning problems. Even when $\lambda = 0$, our proof is new and rather concise (see Appendix).

Theorem 2.1. *Let $\mathbf{w}^*$ be the optimal solution of (1) in $\mathbb{F}$. Assume $\{\mathbf{w}^t\}$ is generated by Smooth SCD. Then*

i) If $P(\mathbf{w}) = \|\mathbf{w}\|_1$, $\phi_t - F(\mathbf{w}^*) \leq O(L/t)$.
ii) If $P(\mathbf{w}) = \|\mathbf{w}\|_2^2$, there exists a $0 < \mu < 1$ such that $\phi_t - F(\mathbf{w}^*) \leq O(\mu^t)$.

By fixing the expected accuracy of solution and the confidence level, we can also derive the same orders of convergence rates of Smooth SCD with high probability as that in [21] and [25]. When $P(\mathbf{w}) = \|\mathbf{w}\|_2^2$, Theorem 2.1 indicates that we have achieved optimal convergence rates that are standard in the literature for strongly convex optimization [18]. If the loss function is smooth but not strongly convex such as L_2-loss, we obtain the convergence rate $O(L/t)$ for its l_1 regularization. However, the rate $O(L/t)$ is not optimal for general convex smooth losses. In a series of work, Nesterov proposed several methods to accelerate convergence of PG. They obtain the optimal convergence rate $O(L/t^2)$ that are standard in the literature [14,15]. As an extension of Nesterov's accelerated method [13], a shrinkage-thresholding Accelerated Proximal Gradient (APG) algorithm was recently proposed in [2]. This accelerated scheme for gradient methods can be done also for the SCD schemes, and several variants that can reach convergence rate $O(L/t^2)$ has been discussed in [18]. Unfortunately for some applications, as pointed out in [18], the complexity of one iteration of the accelerated scheme is rather high since the computation of full-dimensional vectors has to be concerned. As the focus in this paper is only on stochastic algorithms, we will not discuss the accelerated SCD. However, we will compare with the batch APG [2] in the experiments.

A possible drawback of the above scheme especially for stochastic learning is that the Lipschitz constant $L(f)$ is not always known or computable. In the optimization process, $L(f)$ plays a dominant part as the stepsize. The selection of stepsize severely affects the performance of optimization methods even in the stochastic setting. It has been indicated in [26] that SCD is much slower than the corresponding deterministic methods when a too large upper bound of the second derivative is used. This fact indicates that the factor L in convergence rate $O(L/t)$ is extremely useful in practice. We therefore analyze an adaptive SCD algorithm with compact Lipschitz constants in the following.

To ensure the holding of Theorem 2.1, the local Lipschitz condition should be satisfied in each iteration, i.e., for any $\mathbf{w} \in [\mathbf{w}^t, \mathbf{w}^{t+1}]$, there exists a constant L_t such that

$$f(\mathbf{w}) \leq f(\mathbf{w}^t) + \langle \mathbf{w} - \mathbf{w}^t, \nabla f(\mathbf{w}^t) \rangle + (L_t/2)\|\mathbf{w} - \mathbf{w}^t\|^2 \tag{7}$$

with $\inf_{t \geq 1}\{L_t\} > 0$. Obviously, L_t can be roughly selected as an upper bound on the second derivative of the loss. To find a more compact L_t at each step, it is intuitive to use the second-order derivative of f at $\mathbf{w}^t$ as an initial point for line search. Specifically, let $a_i(z) = f(\mathbf{w}^t + z\mathbf{e}_i)$, we solve the following randomly

selected single-variable sub-problem

$$\min_z a_i'(0)z + \frac{1}{2}a_i''(0)z^2 + \lambda p(w_i^t + z) - \lambda p(w_i^t) \tag{8}$$

where $a_i''(0)$ is a subgradient of $a_i'(0)$. To get the solution of (8) in closed-form, we restrict

$$a_i''(0) = \epsilon \quad if \qquad a_i''(0) < \epsilon \tag{9}$$

where ϵ is a predefined sufficiently small positive number. We use the following backtracking line search strategy to find γd,

$$f(\mathbf{w}^t + \gamma d\mathbf{e}_i) - f(\mathbf{w}^t) \leq \gamma[a_i'(0)d + (\gamma/2)d^2] \tag{10}$$

where d is the solution of (8), $\beta \in (1, \infty)$ and $\gamma = \min\{1, \beta, \beta^2, \ldots, \}a_i''(0)$ such that γd satisfies (10). By solving (8) and (10), the update from $\mathbf{w}^t$ to $\mathbf{w}^{t+1}$ is $\mathbf{w}^{t+1} = \mathbf{w}^t + \gamma d\mathbf{e}_i$. We describe our adaptive smooth SCD in Algorithm 2, which is a coordinate-wise version of ISTA with backtracking in [2]. As discussed in [2], Algorithm 2 has the same order of convergence rate as Algorithm 1.

Algorithm 2. Adaptive Smooth SCD

Initialize a weight vector $\mathbf{w}^1$ and choose $\beta \in (1, \infty)$.
repeat
1. Choose $i \in \{1, 2, \ldots, N\}$ uniformly at random
2. calculate $a_i'(0)$ and $a_i''(0)$ according to (9)
3. calculate d via (8)
4. compute $\gamma = \min\{1, \beta, \beta^2, \ldots, \}a_i''(0)$ such that γd satisfies (10) and update $\mathbf{w}^t$
5. $t := t + 1$

until a stopping condition is satisfied

In [20], an $O(L/t)$ convergence rate for SCD was indeed obtained, but it didn't discuss the linear convergence rate for strongly convex objective functions and only focused on several specific data sets with fixed Lipschitz constants. In [26], a practical CD method using one-dimensional Newton direction (CDN) to minimize the second-order approximation in (2) was proposed. CDN uses the line search strategy (3) to find the parameter γ and the optimal linear convergence rate has been obtained when dealing with strongly convex objective functions (see also [3,6]), but its convergence rate for L_2-loss with l_1 regularizer was not explicitly described in [26]. At first sight, our line search strategy (10) is only a little different from the strategy (3). Nevertheless, our strategy looks rather natural and flexible. Further, the principles behind (10) and (3) are quite different, i.e., the goal of the former is to guarantee the effectiveness of a local Lipschitz expansion and decrease the factor in convergence rates while that of the latter is only to ensure sufficient decrease of the objective function. At this point, we have established a close link between the practical CD algorithms in [24,26] and principled structural optimization techniques. Note that (10) has the

same computational cost as (3), thus we believe that SCD algorithms developed in this section can fill the gap between convergence rate analysis and practical efficiency. Compared with the SCD in [20] and CDN in [26], our method is interesting in both theory and practice.

3 SCD Algorithms for Nonsmooth Losses

In this section, we only assume that f is only continuous and convex. Since the generalized second derivative of f doesn't exist, we can not discuss CD algorithms along the lines in Section 2. This is the main obstacle to establish CD algorithms for nonsmooth losses. Note that many SCD algorithms as well as their convergence heavily depend on the associated corresponding full-gradient algorithms [18,20]. This fact motivates us to start from the nonsmooth deterministic optimization method.

In [17], a dual averaging method was presented for different types of nonsmooth problems only requiring the subgradient information. At each iteration, the learning variables are adjusted by solving a simple minimization problem that involves running average of all past subgradients that emphasizes more recent gradients. RDA is an extension of this method, which can solve online regularized learning problems [25]. More specifically, the key iteration of batch RDA takes the form

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} \left\{ \langle \bar{\mathbf{g}}^t, \mathbf{w} \rangle + \lambda P(\mathbf{w}) + (\beta_t/t) h(\mathbf{w}) \right\} \tag{11}$$

where $\bar{\mathbf{g}}^t = \frac{1}{t} \sum_{j=1}^{t} \nabla f(\mathbf{w}^j)$, $\nabla f(\mathbf{w}^j)$ is a subgradient of f at $\mathbf{w}^j$, $h(\mathbf{w})$ is an auxiliary strongly convex function, and $\{\beta_t\}_{t \geq 1}$ is a nonnegative and nondecreasing input sequence. We describe RDA for batch learning in Algorithm 3.

Algorithm 3. Batch RDA

Initialize a weight vector $\mathbf{w}^1 = \mathbf{0}$ and $\bar{\mathbf{g}}^0 = \mathbf{0}$.
repeat
1. compute $\mathbf{g}^t = \nabla f(\mathbf{w}^t)$
2. update $\bar{\mathbf{g}}^t = [(t-1)\bar{\mathbf{g}}^{t-1} + \mathbf{g}^t]/t$
3. compute $\mathbf{w}^{t+1}$ via (11)
4. $t := t + 1$

until a stopping condition is satisfied

For simplicity, we choose $h(\mathbf{w}) = (1/2)\|\mathbf{w}\|_2^2$ throughout this paper. According to (11), $\mathbf{w}^{t+1}$ can be found in closed-form with little effort. This is the main reason that RDA methods can successfully deal with large scale problems and exploit the regularization structure. As optimization problem (11) is separable, we can get $\mathbf{w}^{t+1}$ by solving each w_i^{t+1} independently, i.e.,

$$w_i^{t+1} = \arg\min_{\omega} \left\{ \omega \bar{g}_i^t + \lambda p(\omega) + (\beta_t/2t)\omega^2 \right\} \tag{12}$$

where $\bar{g}_i^t$ denotes the i-th component of $\frac{1}{t}\sum_{j=1}^{t}\nabla f(\mathbf{w}^j)$.

To conduct convergence analysis, we gather the following assumptions from [25], although some of them don't appear explicitly in Theorem 3.1 and 3.2.

Assumption 3.1. *There exists a constant* $M > 0$ *such that* $\|\nabla_i f(\mathbf{w})\| \leq M$, $\forall \mathbf{w} \in \mathbb{R}^N$, $1 \leq \forall i \leq N$ and $\max\{\sigma_1, \beta_1\} > 0$, where σ_1 is dedicated to the convexity parameter of $P(\mathbf{w})$. If $P(\mathbf{w}) = \|\mathbf{w}\|_1$, β_t is order exactly $\sqrt{t}$. If $P(\mathbf{w}) = \|\mathbf{w}\|_2^2$, $\beta_t \leq O(\ln t)$.

[25] gave several precise regret bounds of the RDA method for solving regularized online problems. The convergence rates for stochastic learning problems can be established based on these regret bounds. If the regularizer is general convex such as $P(\mathbf{w}) = \|\mathbf{w}\|_1$, the online RDA has an $O(\sqrt{t})$ regret bound. If the regularization term is strongly convex such as $P(\mathbf{w}) = \|\mathbf{w}\|_2^2$, the online RDA has an $O(\ln t)$ regret bound. As a direct consequence of regret analysis in [25], we can get

Theorem 3.1. *Let* $\mathbb{F}_D = \{h(\mathbf{w}) \leq D^2\}$ *and* $\mathbf{w}^*$ *be the optimal solution of* (1) *in* $\mathbb{F}_D$. *Assume* $\{\mathbf{w}^t\}$ *is generated by Batch RDA and* $\bar{\mathbf{w}}^t = \frac{1}{t}\sum_{j=1}^{t}\mathbf{w}^j$. *Then*

i) If $P(\mathbf{w}) = \|\mathbf{w}\|_1$, $F(\bar{\mathbf{w}}^t) - F(\mathbf{w}^*) \leq O(\frac{MD\sqrt{t}}{t})$.

ii) If $P(\mathbf{w}) = \|\mathbf{w}\|_2^2$, $F(\bar{\mathbf{w}}^t) - F(\mathbf{w}^*) \leq O(\frac{(2D^2+\frac{M^2}{4})(1+lnt)}{t})$.

In order to derive CD algorithms for regularized nonsmooth losses, we at each step only solve (11) on one component which is randomly selected from total N components instead of separately going over all the components in the batch setting. In particular, if d is the solution of the randomly selected one-variable subproblem in the form of (12), the update from $\mathbf{w}^t$ to $\mathbf{w}^{t+1}$ becomes $\mathbf{w}^{t+1} = \mathbf{w}^t + d\mathbf{e}_i$. We describe our primal SCD algorithm for nonsmooth losses in Algorithm 4.

Algorithm 4. Nonsmooth SCD

Initialize a weight vector $\mathbf{w}^1 = \mathbf{0}$ and $\bar{\mathbf{g}}^0 = \mathbf{0}$.
repeat
1. Choose $i \in \{1, 2, \ldots, N\}$ uniformly at random
2. let g_i^t be the i-th element of $\nabla f(\mathbf{w}^t)$
3. update $\bar{g}_i^t = [(t-1)\bar{g}_i^{t-1} + g_i^t]/t$
4. solve (12) and update $\mathbf{w}^{t+1}$
5. $t := t + 1$

until a stopping condition is satisfied

To measure the stochastic quality of the solutions $\mathbf{w}^1, \ldots, \mathbf{w}^t$, we first prove (in Appendix)

Lemma 3.1. *Assume* $\mathbf{w}^t$ *is generated by Nonsmooth CD.* $\forall \mathbf{w} = (w_1, w_2, \ldots, w_N)^T \in \mathbb{F}_D$, *define* $\delta_t(\mathbf{w}) = \sum_{\tau=1}^{t}\{g_{i_\tau}^\tau(w_{i_\tau}^\tau - w_{i_\tau}) + \lambda p(w_{i_\tau}^\tau)\} - \frac{1}{N}t\lambda P(\mathbf{w})$ *and* $R_t(\mathbf{w}) = \sum_{\tau=1}^{t}\{\boldsymbol{\phi}_\tau - F(\mathbf{w})\}$. *Then* $R_t(\mathbf{w}) \leq N\mathbf{E}_{\xi_t}\delta_t(\mathbf{w})$

From the proof of Lemma 3.1,$N\mathbf{E}_{\xi_t}\delta_t(\mathbf{w}) = \mathbf{E}_{\xi_t}\sum_{\tau=1}^{t}[\mathbf{g}^\tau(\mathbf{w}^\tau - \mathbf{w}) + \lambda P(\mathbf{w}^\tau) - \lambda P(\mathbf{w})]$. In online learning, the bound of $\sum_{\tau=1}^{t}\{\mathbf{g}^\tau(\mathbf{w}^\tau - \mathbf{w}) + \lambda P(\mathbf{w}^\tau) - \lambda P(\mathbf{w})\}$ has been analyzed in [25]. The regret, primal variable and dual average can be bounded based on this bound. Following similar arguments, we can derive

Theorem 3.2. *Let* $\mathbf{w}^*$ *be the optimal solution of (1) in* $\mathbb{F}_D$. *Assume* $\{\mathbf{w}^t\}$ *is generated by Stochastic Nonsmooth CD. Then*

i) If $P(\mathbf{w}) = \|\mathbf{w}\|_1$, $\frac{1}{t}\sum_{\tau=1}^{t}\boldsymbol{\phi}_\tau - F(\mathbf{w}^*) \leq O(\frac{MD\sqrt{t}}{t})$.

ii) If $P(\mathbf{w}) = \|\mathbf{w}\|_2^2$, $\frac{1}{t}\sum_{\tau=1}^{t}\boldsymbol{\phi}_\tau - F(\mathbf{w}^*) \leq O(\frac{(2D^2+\frac{M^2}{4})(1+lnt)}{t})$.

In addition to convergence in expectation, we can also derive the same orders of convergence rates with high probability. Theorem 3.2 indicates that we have achieved the optimal convergence rates that are standard in the literature for convex and strongly convex nonsmooth losses minimization. At first sight, online RDA in [25] and our Nonsmooth SCD share the same idea in principle, i.e., both of them approach the solution of (1) by optimizing the regularized dual averaging objective function defined in [17]. However, the former accomplishes the task of sparse online learning and the latter expands the field of SCD algorithms.

4 Experiments

In this section, we will present experiments to validate our theoretical analysis and demonstrate the performance of our algorithms. Typically, we consider one toy data set and four large scale data sets. The toy data set with 800 samples in $\mathbb{R}^{800}$ is generated like [4], i.e., we choose a $\mathbf{w}$ with entries distributed normally with 0 mean and unit variance and randomly zeroed 50% of the vector, the data matrix $\mathbf{X} \in \mathbb{R}^{800\times 800}$ was random with entries also normally distributed, and we set $\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{v}$, where the components of $\mathbf{v}$ were also distributed normally at random. The four real data sets are described in Table 1. We do not include the bias term for all the solvers. All algorithms are implemented in C++ and all the experiments are run on a Sun Ultra 45 Workstation with 1.6GHz UltraSPARC IIIi processor and 4GB of main memory under Solaris 10. The trade-off parameter λ is chosen by using the cross validation strategy. To have a fair comparison, each stochastic algorithm is run 10 times and the reported are averaged results. We find that SCD achieves consistently better test accuracy than other solvers. For clarity, we category the experiments into nonsmooth and smooth loss problems.

Nonsmooth Loss Problems. We first consider the hinge loss with l_1 regularizer (l_1-R-L_1) problem. It has been shown in [4] that the variants of stochastic gradient projection methods augmented with L1 efficient projection procedures outperform many optimization techniques such as exponentiated gradient algorithm. Specifically, we can employ the efficient projection algorithm in [4] to implement the PSA for hinge loss (L1-PSA) ([22]). Since few papers study large scale l_1-R-L_1 problems and they are excluded from the comparison in [26], to illustrate the scalability of our Nonsmooth SCD, we choose to compare

Table 1. Real Data-sets where the split describes the size of a train/test set

Data-set	Dimension	Split
Astro-Physics	99,757	29,882/32,487
CCAT	47,236	23,149/781,265
A9a	123	24,703/7,858
Covtype	54	522,911/58,101

our Nonsmooth SCD with L1-PSA and Batch RDA. In the experiments, Nonsmooth SCD obtains the same level of sparsity as Batch RDA. The relationship between$|(F(\mathbf{w}^t) - F(\mathbf{w}^*)|/|F(\mathbf{w}^*)|$ vs. CPU time is illustrated in Fig. 1, Fig. 2, Fig. 3 and Fig. 4.

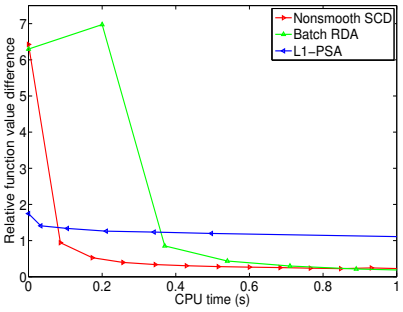


Fig. 1. l_1-R-L_1 on Astro-ph

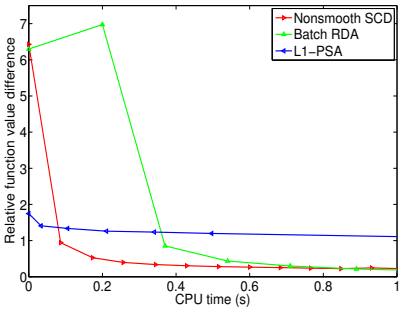


Fig. 2. l_1-R-L_1 on CCAT

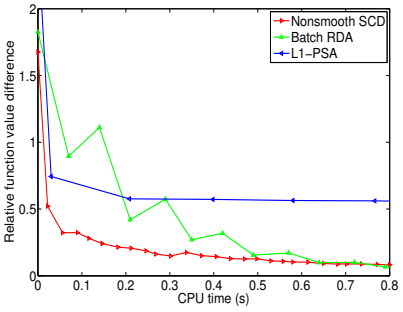


Fig. 3. l_1-R-L_1 on A9a

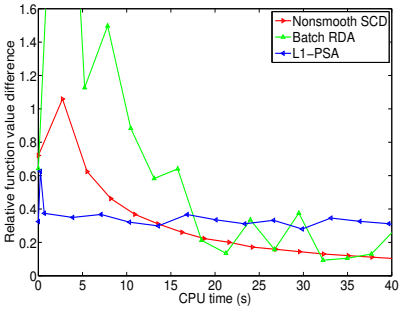


Fig. 4. l_1-R-L_1 on Covertype

We then consider the hinge loss with l_2 regularizer (l_2-R-L_1). For this problem, one of the most efficient primal algorithms is Pegasos in [21] and the state-of-the-art dual algorithm is Dual SCD in [6]. In particular, the experiments in [6] indicate that Dual CD is much faster than many solvers such as Pegasos, TRON [9], SVM$^{\text{perf}}$ [7]. To illustrate the scalability of our Nonsmooth SCD, we choose to compare with Pegasos ($\lambda_{\text{pegasos}} = 2\lambda_{\text{SCD}}/m$, [6]), Dual SCD and Batch RDA.

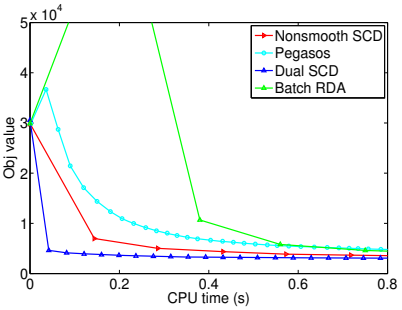


Fig. 5. l_2-R-L_1 on Astro-ph

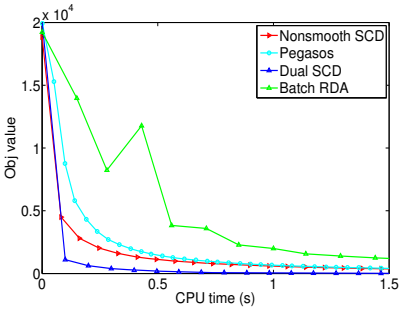


Fig. 6. l_2-R-L_1 on CCAT

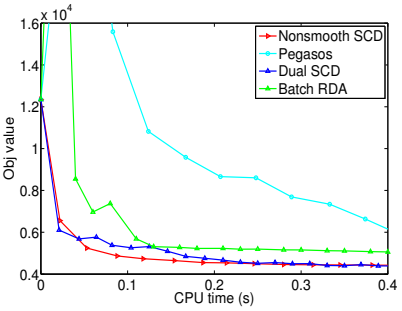


Fig. 7. l_2-R-L_1 on A9a

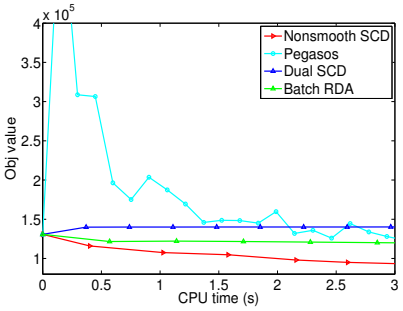


Fig. 8. l_2-R-L_1 on Covertype

The relationship between $F(\mathbf{w}^t) - F(\mathbf{w}^*)$ vs. CPU time is illustrated in Fig. 5, Fig. 6, Fig. 7 and Fig. 8.

In l_1 and l_2 regularized experiments, three kinds of phenomena are observed: 1) our Nonsmooth SCD converges faster than Batch RDA. 2) our primal Nonsmooth SCD is faster than the two other primal algorithms L1-PSA and Pegasos. 3) the performance of primal CD degrades as the datasets get larger and this can be seen when the size of the training set is greater than the dimension ($m < N$). For example, on Astro-physics and CCAT, our primal Nonsmooth SCD is a little slower than the state of the art Dual SCD in [6] (Fig. 5 and 6). 4) the performance of primal CD upgrade as the dimensions get larger and this can be seen when the size of the training set is less than the dimension ($m > N$). For example, our primal Nonsmooth SCD has similar performance as the state of the art Dual on CD A9a and outperforms it on Convertype (Fig. 7 and 8).

Based the above experimental results, SCD methods outperform their corresponding deterministic approaches and Nonsmooth SCD is among the first optimization schemes suggested for efficiently solving large scale nonsmooth primal learning problems. We conclude that our Nonsmooth SCD has achieved all the expected effects that a primal SCD algorithm should have.

Smooth Loss Problems. Our Adaptive Smooth SCD algorithm can deal with many learning problems such as the popular regularized squared and logistic loss

problems considered in [20]. To illustrate our main contribution in smooth cases, we only consider L_2-loss.

In [26], many scalable algorithms for regularized smooth losses were compared. These solvers include CDN, SCD [20], CGDGS [27], IPM [8], Lassplore [10] and GLMNET [5]. The extensive experiments sufficiently illustrate that CDN is the fastest. Therefore, to illustrate the scalability of our Adaptive Smooth SCD, we only focus on comparing with the stochastic CDN in [26]. The relationships between $F(\mathbf{w}^t)$ and CPU time are illustrated in Fig. 9, Fig. 10, Fig. 11 and Fig. 12. In addition to obtaining the same level of both sparsity and test accuracy, from these figures, it is easy to find that our Adaptive smooth SCD has almost the same practicality as the stochastic CDN in [26].

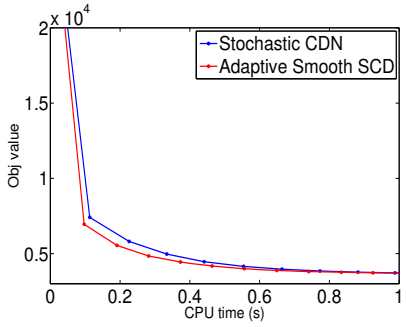


Fig. 9. l_1-R-L_2 on Astro-ph

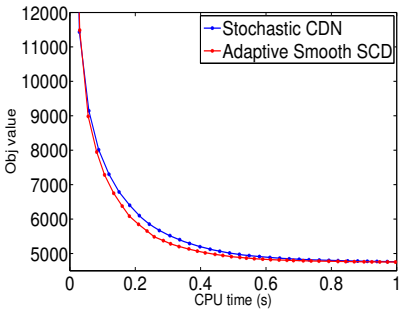


Fig. 10. l_1-R-L_2 on CCAT

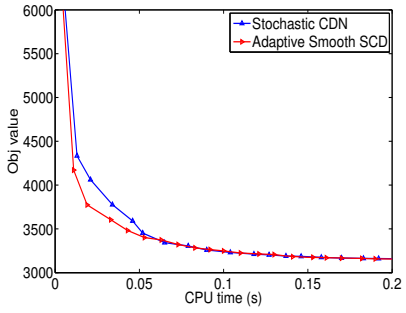


Fig. 11. l_1-R-L_2 on A9a

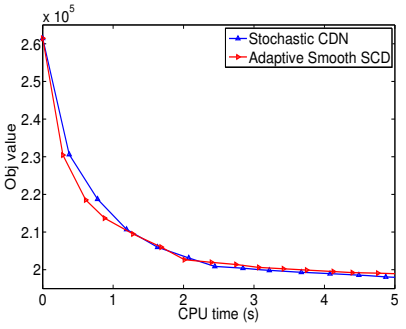


Fig. 12. l_1-R-L_2 on Covertype

To further illustrate the effectiveness of our line search strategy in (10), we also do a toy experiment. One purpose is to compare the performance of PG, APG and Adaptive Smooth SCD when the global Lipschitz constant of ∇f is known, and the other is to test if our line search strategy (10) can really select a more aggressive local Lipschitz constant.

On the toy data set, we follow the strategy in [11] and [12] to calculate the global Lipschitz constant of ∇f. We use this Lipschitz constant to implement PG

and APG. The objective values vs. CPU time are illustrated in Fig. 13. From Fig. 13, we can see that our Adaptive Smooth SCD converges much faster than PG. This fact shows that SCD methods with line search strategy outperform their corresponding deterministic approaches in smooth loss cases. What is more, our Adaptive Smooth SCD converges even faster than APG.

More details about the selection of Lipschitz constant are reported in Fig. 14, where the red points represent the local componentwise Lipschitz constant in each iteration of Adaptive Smooth SCD while the blue line is the global Lipschitz constant. From Fig. 14, we find that the local componentwise Lipschitz constant is much smaller than the global Lipschitz constant. This fact means that our line search strategy can decrease the factor of the convergence rates and then improve the performance of smooth SCD. Based on our theoretic analysis

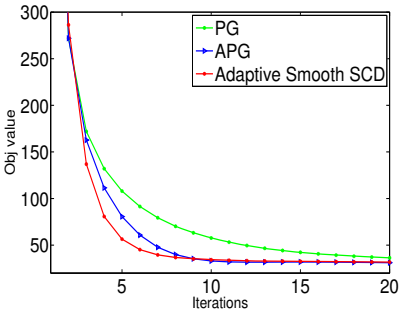


Fig. 13. l_1-R-L_2 on toy data set

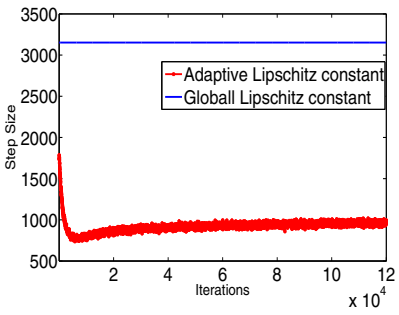


Fig. 14. The adaptive Lipschitz constant

in Section 3 and experimental results in this example, we conclude that our Adaptive Smooth SCD has achieved the expected effects in both convergence rates and practicality. Therefore, Adaptive Smooth SCD is a principled and practical method for solving large scale problems.

5 Conclusion

In this paper, we have established an interesting framework for developing SCD algorithms for regularized both nonsmooth and smooth losses minimization from structural optimization techniques. We have analyzed how our algorithms are not worse than the state-of-the-art scalable solvers. Experiments confirm the correctness of our theoretical analysis and efficiency of the proposed algorithms. There are several possible extension to this work. For example, the cyclic CD algorithms for regularized nonsmooth losses and the comparison analysis of runtime bounds. These will be included in our future work.

Acknowledgments. The work was supported in part by the NSFC (Grant No. 60835002, 60975040 and 61175050) and the first author is also supported by the Open Project Program of the NLPR. We thank the anonymous reviewers for their helpful comments.

Appendix

To prove Theorem 2.1, we first give the following key lemma [23].
3-Point Property. *Assume*

$$\mathbf{w}^{\tau+1} = \arg\min_{\mathbf{w}} l_f(\mathbf{w}, \mathbf{w}^\tau) + \lambda P(\mathbf{w}) + \tfrac{L}{2}\|\mathbf{w} - \mathbf{w}^\tau\|^2$$

Then $\forall \mathbf{w} \in \mathbb{R}^N$, *we have*

$$l_f(\mathbf{w}, \mathbf{w}^\tau) + \lambda P(\mathbf{w}) + \tfrac{L}{2}\|\mathbf{w} - \mathbf{w}^\tau\|^2$$
$$\geq l_f(\mathbf{w}^{\tau+1}, \mathbf{w}^\tau) + \lambda P(\mathbf{w}^{\tau+1}) + \tfrac{L}{2}\|\mathbf{w}^{\tau+1} - \mathbf{w}^\tau\|^2 + \tfrac{L}{2}\|\mathbf{w} - \mathbf{w}^{\tau+1}\|^2$$

Proof of Theorem 2.1

i) Note

$$\boldsymbol{\phi}_\tau = \mathbf{E}_{\xi_\tau} F(\mathbf{w}^{\tau+1}) = \mathbf{E}_{\xi_{\tau-1}} \mathbf{E}_{i_\tau} F(\mathbf{w}^{\tau+1})$$

$\forall \mathbf{w} \in \mathbb{R}_N$, by using the smooth assumption

$$\mathbf{E}_{i_\tau} F(\mathbf{w}^{\tau+1}) \leq \mathbf{E}_{i_\tau}[l_f(\mathbf{w}^{\tau+1}, \mathbf{w}^t) + \lambda P(\mathbf{w}^{\tau+1}) + \tfrac{L}{2}\|\mathbf{w}^{\tau+1} - \mathbf{w}^\tau\|^2]$$

By using the 3-Point Property,

$$\mathbf{E}_{i_\tau} F(\mathbf{w}^{\tau+1}) \leq \mathbf{E}_{i_\tau}[l_f(\mathbf{w}, \mathbf{w}^\tau) + \lambda P(\mathbf{w}) + \tfrac{L}{2}\|\mathbf{w} - \mathbf{w}^\tau\|^2 - \tfrac{L}{2}\|\mathbf{w} - \mathbf{w}^{\tau+1}\|^2]$$
$$\leq F(\mathbf{w}) + \tfrac{L}{2}\|\mathbf{w} - \mathbf{w}^\tau\|^2 - \tfrac{L}{2}\mathbf{E}_{i_\tau}\|\mathbf{w} - \mathbf{w}^{\tau+1}\|^2$$

So,

$$\boldsymbol{\phi}_t \leq \mathbf{E}_{\xi_{\tau-1}}[F(\mathbf{w}) + \tfrac{L}{2}\|\mathbf{w} - \mathbf{w}^\tau\|^2 - \tfrac{L}{2}\mathbf{E}_{i_\tau}\|\mathbf{w} - \mathbf{w}^{\tau+1}\|^2]$$
$$\leq F(\mathbf{w}) + \tfrac{L}{2}\mathbf{E}_{\xi_{\tau-1}}\|\mathbf{w} - \mathbf{w}^\tau\|^2 - \tfrac{L}{2}\mathbf{E}_{\xi_\tau}\|\mathbf{w} - \mathbf{w}^{\tau+1}\|^2$$

Adding the above inequalities from $\tau = 1$ to $\tau = t$,

$$\sum_{\tau=1}^{t}[\boldsymbol{\phi}_\tau - F(\mathbf{w})] \leq \tfrac{L}{2}\|\mathbf{w} - \mathbf{w}^1\|^2$$

On the other hand,

$$\boldsymbol{\phi}_\tau - F(\mathbf{w}) \leq \boldsymbol{\phi}_{\tau-1} - F(\mathbf{w}),\ t[\boldsymbol{\phi}_t - F(\mathbf{w})] \leq \sum_{\tau=1}^{t}[\boldsymbol{\phi}_\tau - F(\mathbf{w})] \leq \tfrac{L}{2}\|\mathbf{w} - \mathbf{w}^1\|^2$$

This proves i) in Theorem 2.1.

ii) If $P(\mathbf{w}) = \|\mathbf{w}\|_2^2$, F becomes a strongly convex function with Lipschiz constant $L + 2\lambda$. This fact implies that ii) follows from Theorem 2 in [18].

Proof of Lemma 3.1

$$\mathbf{E}_{\xi_t} \sum_{\tau=1}^{t} \{g_{i_\tau}^{\tau}(w_{i_\tau}^{\tau} - w_{i_\tau}) + \lambda p(w_{i_\tau}^{\tau})\} = \sum_{\tau=1}^{t} \mathbf{E}_{\xi_\tau} \{g_{i_\tau}^{\tau}(w_{i_\tau}^{\tau} - w_{i_\tau}) + \lambda p(w_{i_\tau}^{\tau})\}$$

$$= \sum_{\tau=1}^{t} \mathbf{E}_{\xi_{\tau-1}} \mathbf{E}_{i_\tau} \{g_{i_\tau}^{\tau}(w_{i_\tau}^{\tau} - w_{i_\tau}) + \lambda p(w_{i_\tau}^{\tau})\}$$

By the definition of expectation in i_τ, we obtain

$$\mathbf{E}_{i_\tau} \{g_{i_\tau}^{\tau}(w_{i_\tau}^{\tau} - w_{i_\tau}) + \lambda p(w_{i_\tau}^{\tau})\} = \tfrac{1}{N}[\mathbf{g}^{\tau}(\mathbf{w}^{\tau} - \mathbf{w}) + \lambda P(\mathbf{w}^{\tau})]$$

According to the definition of subgradient $\langle \mathbf{g}^{\tau}, \mathbf{w}^{\tau} - \mathbf{w} \rangle \geq f(\mathbf{w}^{\tau}) - f(\mathbf{w})$. So,

$$\tfrac{1}{N}[\mathbf{g}^{\tau}(\mathbf{w}^{\tau} - \mathbf{w}) + \lambda P(\mathbf{w}^{\tau})] - \tfrac{1}{N}\lambda P(\mathbf{w}) \geq \tfrac{1}{N}[f(\mathbf{w}^{\tau}) - f(\mathbf{w}) + \lambda P(\mathbf{w}^{\tau})] - \tfrac{1}{N}\lambda P(\mathbf{w})$$

$$= \tfrac{1}{N}[F(\mathbf{w}^{\tau}) - F(\mathbf{w})]$$

By taking the expectation $\mathbf{E}_{\xi_{\tau-1}}$,

$$\mathbf{E}_{\xi_{\tau-1}} \mathbf{E}_{i_\tau} \{g_{i_\tau}^{\tau}(w_{i_\tau}^{\tau} - w_{i_\tau}) + \lambda p(w_{i_\tau}^{\tau})\} - \tfrac{1}{N}\lambda P(\mathbf{w})$$

$$\geq \tfrac{1}{N} \mathbf{E}_{\xi_{\tau-1}}[F(\mathbf{w}^{\tau}) - F(\mathbf{w})] \geq \tfrac{1}{N}[\phi^{\tau} - F(\mathbf{w})]$$

By adding the above inequalities from $\tau = 1$ to $\tau = t$, the lemma is proved.

References

1. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University Press (2004)
2. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences 2(1), 183–202 (2009)
3. Chang, K.W., Hsieh, C.J., Lin, C.J.: Coordinate descent method for large-scale L_2-loss linear support vector machines. Journal of Machine Learning Research 9, 1369–1398 (2008)
4. Duchi, J., Shalev-Shwartz, S., Singer, Y., Chandra, T.: Efficient projections onto the l 1-ball for learning in high dimensions. In: Proceedings of the 25th International Conference on Machine Learning, pp. 272–279 (2008)
5. Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33(1), 1–22 (2010)
6. Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the 25th International Conference on Machine Learning, pp. 408–415 (2008)
7. Joachims, T.: Training linear SVMs in linear time. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 217–226 (2006)
8. Koh, K., Kim, S.J., Boyd, S.: An interior-point method for large-scale l_1-regularized logistic regression. Journal of Machine Learning Research 8, 1519–1555 (2007)
9. Lin, C.J., Weng, R.C., Keerthi, S.S.: Trust region newton method for logistic regression. Journal of Machine Learning Research 9, 627–650 (2008)
10. Liu, J., Ye, J.: Efficient Euclidean projections in linear time. In: Proceedings of the 26th International Conference on Machine Learning, pp. 657–664 (2009)
11. Mangasarian, O.L.: A finite Newton method for classification. Optimization Methods and Software 17(5), 913–929 (2002)
12. Mangasarian, O.L., Musicant, D.R.: Successive overrelaxation for support vector machines. IEEE Trans. Neural Networks 10, 1032–1037 (1999)

13. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Soviet Mathematics Doklady 27, 372–376 (1983)
14. Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical Programming 103(1), 127–152 (2005)
15. Nesterov, Y.: Gradient methods for minimizing composite objective function. CORE Discussion Papers 2007076. Center for Operations Research and Econometrics, CORE (2007)
16. Nesterov, Y.: How to advance in structural convex optimization. OPTIMA: Mathematical Programming Society Newsletter 78, 2–5 (2008)
17. Nesterov, Y.: Primal-dual subgradient methods for convex problems. Mathematical Programming 120(1), 221–259 (2009)
18. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. Technical report, University catholique de Louvain, Center for Operations Research and Econometrics, CORE (2010)
19. Saha, A., Tewari, A.: On the finite time convergence of cyclic coordinate descent methods. Arxiv preprint arXiv:1005.2146 (2010)
20. Shalev-Shwartz, S., Tewari, A.: Stochastic methods for l_1 regularized loss minimization. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 929–936 (2009)
21. Shalev-Shwartz, S., Singer, Y., Srebro, N.P.: Primal estimated sub-gradient solver for SVM. In: Proceedings of the 24th International Conference on Machine Learning, pp. 807–814 (2007)
22. Tao, Q., Sun, Z., Kong, K.: Developing Learning Algorithms via Optimized Discretization of Continuous Dynamical Systems. IEEE Trans. Syst. Man Cybern. B 42(1), 140–149 (2012)
23. Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization. submitted to SIAM Journal on Optimization (2008)
24. Tseng, P., Yun, S.: A coordinate gradient descent method for nonsmooth separable minimization. Mathematical Programming 117(1), 387–423 (2009)
25. Xiao, L.: Dual averaging methods for regularized stochastic learning and online optimization. Journal of Machine Learning Research 11, 2543–2596 (2010)
26. Yuan, G.X., Chang, K.W., Hsieh, C.J., Lin, C.: A Comparison of Optimization Methods and Software for Large-scale L_1-regularized Linear Classification. Journal of Machine Learning Research 11, 3183–3234 (2010)
27. Yun, S., Toh, K.C.: A coordinate gradient descent method for l_1-regularized convex minimization. To appear in Computational Optimizations and Applications (2009)

Sublinear Algorithms for Penalized Logistic Regression in Massive Datasets

Haoruo Peng[1,2], Zhengyu Wang[1,3], Edward Y. Chang[1], Shuchang Zhou[1], and Zhihua Zhang[1,4]

[1] Google Research Beijing, Beijing, China 100084
[2] Department of Computer Science and Technology Tsinghua University, Beijing, China 100084
[3] Institute for Interdisciplinary Information Sciences Tsinghua University, Beijing, China 100084
[4] College of Computer Science and Technology Zhejiang University, Zhejiang, China 310027
penghaoruo@hotmail.com, wangsincos@163.com, eyuchang@gmail.com, georgezhou@google.com, zhzhang@cs.zju.edu.cn

Abstract. Penalized logistic regression (PLR) is a widely used supervised learning model. In this paper, we consider its applications in large-scale data problems and resort to a stochastic primal-dual approach for solving PLR. In particular, we employ a random sampling technique in the primal step and a multiplicative weights method in the dual step. This technique leads to an optimization method with sublinear dependency on both the volume and dimensionality of training data. We develop concrete algorithms for PLR with ℓ_2-norm and ℓ_1-norm penalties, respectively. Experimental results over several large-scale and high-dimensional datasets demonstrate both efficiency and accuracy of our algorithms.

1 Introduction

The penalized logistic regression (PLR) model [9] plays an important role in machine learning and data mining. The model serves for classification problems, and enjoys a substantial body of supporting theories and algorithms. PLR is competitive with the support vector machines (SVMs) [18], because it has both high accuracy and interpretability (PLR can directly estimate a conditional class probability).

Recently, large-scale applications have emerged from many modern massive datasets. A key characteristic of these applications is that the size of their training data is very large and data dimensionality is very high. For example, in medical diagnostic applications [17], both doctors and patients would like to take the advantage of millions of records over hundreds of attributes. More evidently, search engines on texts or multimedia data must handle data volume in the billion scale and each data instance is characterized by a feature space of thousands of dimensions [7]. Large data volume and high data dimensionality pose computational challenges to machine learning problems.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 553–568, 2012.

In this paper, we tackle these challenges via stochastic approximation approaches. Stochastic approximation methods, such as stochastic gradient descent [20] and stochastic dual averaging [19], obtain optimal generalization guarantees with only a single pass or a small number of passes over the data. Therefore, they can achieve a desired generalization with runtime linear to the dataset size. We further speed up the runtime, and propose sublinear algorithms for PLR via the use of stochastic approximation idea. Our algorithms work at the same level of performance with traditional learning methods for PLR, but require much shorter running time. Our methods access a single feature of training vectors instead of entire training vectors at each iteration. This *sampling* approach brings much improved computational efficiency by eliminating a large number of vector multiplication operations. By devising clever randomized algorithms, we can also enjoy the benefits of taking less number of iterations and hence accessing less number of features. Such reduction in accessing features can substantially reduce running time as pointed out by [11].

Our algorithms can be easily applied to distributed storage systems [12] with parallel updates on all instances. Compared with other traditional batch algorithms, we do not require any global reduction [14] computation, which is a speedup bottleneck. Thus, our algorithms can achieve significant speedup on massive datasets.

The rest of the paper is organized as follows: Section 2 discusses some related work. In Section 3, we review some preliminaries and explain the setting along with the model. In Section 4, we present the framework of our sublinear algorithms for PLR. In Section 5, we depict detailed algorithms and analysis. Section 6 describes the datasets and the baseline of our experiments and presents the experimental results. Finally, we offer our concluding remarks in Section 7.

2 Related Work

There are many existing techniques that address logistic regression with ℓ_1-penalty in the literature.

The *Reduced Memory Multi-pass* (RMMP) algorithm, proposed by Balakrishnan and Madigan [2], is one of the most accurate and fastest convergent algorithms. RMMP trains sparse linear classifiers on high-dimensional datasets in a multi-pass manner. However, this algorithm has computational complexity and memory requirements that make learning on large-scale datasets infeasible. The central idea of the work is a straightforward quadratic approximation to the likelihood function. When the dimensionality of the data gets large, the cost of many vector-vector multiplication operations increases significantly. Also, the quadratic approximation is added together for all instances in each iteration, and such computation inevitably requires global reduction in a distributed storage system.

The *Hybrid Iterative Shrinkage* (HIS) algorithm, proposed by Shi et al. [15], is also computationally efficient without loss of classification accuracy. This algorithm includes a fixed point continuation phase and an interior point phase. The

first phase is based completely on memory efficient operations such as matrix-vector multiplications, while the second phase is based on a truncated Newton's method. Thus, HIS is in the scope and constraints of traditional way of solving the optimization problem. As RMMP has relatively better scalability and performance, we choose to use RMMP instead of HIS as our baseline for the empirical comparison in this paper.

Recently, Clarkson et al. [3] proposed a new method by taking advantage of randomized algorithms. They presented sublinear-time approximation algorithms for optimization problems arising in machine learning, such as linear classifiers and minimum enclosing balls. The algorithm uses a combination of a novel sampling techniques and a new multiplicative update algorithm. They also proved lower bounds which show the running times to be nearly optimal on the unit-cost RAM model.

Hazan et al. [11] exploited sublinear approximation approach to the linear SVM with ℓ_2-penalty, from which we were inspired and borrowed some of the ideas (We generally refer to them as the ETN framework in Section 4). Later on, Cotter et al. [4] extended the work to kernelized SVM cases. In [10], Hazan et al. applied the sublinear approximation approach for solving ridge (ℓ_2-regularized) and lasso (ℓ_1-regularized) linear regression. Garber and Hazan [6] developed the method in semidenfinite programming (SDP).

3 Penalized Logistic Regression Models

Logistic regression is a widely used method for solving classification problems. In this paper, we are mainly concerned with the binary classification problem. Suppose that we are given a set of training data $\mathcal{X} = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ are input samples and $y_i \in \{-1, 1\}$ are the corresponding labels. For simplicity, we let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$. In the logistic regression model, the expected value of y_i is given by

$$P(y_i|\mathbf{x}_i) = \frac{1}{1 + \exp(-y_i(\mathbf{x}_i^T\mathbf{w} + b))} \triangleq g_i(y_i),$$

where $\mathbf{w} = (w_1, \ldots, w_d)^T \in \mathbb{R}^d$ is a regression vector and $b \in \mathbb{R}$ is an offset term. The log likelihood function $F(\mathbf{w}, b; \mathcal{X})$ on the training data is given as

$$F(\mathbf{w}, b|\mathcal{X}) = \sum_{i=1}^{n} \log g_i(y_i).$$

Under the penalized framework, one imposes a prior $p(\mathbf{w})$ to $\mathbf{w}$. This allows us to address the maximum a posteriori (MAP) estimation for $\mathbf{w}$ as

$$\max_{\mathbf{w},b} \big\{ \log p(\mathbf{w}, b|\mathcal{X}) \propto F(\mathbf{w}, b|\mathcal{X}) + \log p(\mathbf{w}) \big\}. \tag{1}$$

In this paper, we consider Gaussian and Laplace priors for $\mathbf{w}$, which in turn induce the ℓ_2 and ℓ_1 penalties for $\mathbf{w}$, respectively.

3.1 The ℓ_2-Penalty Logistic Regression

We assume that $\mathbf{w}$ follows a Gaussian distribution with mean $\mathbf{0}$ and covariance matrix $\lambda\mathbf{I}_d$ where $\mathbf{I}_d$ is the $d\times d$ identity matrix, i.e. $\mathbf{w} \sim N(\mathbf{0}, \lambda\mathbf{I}_d)$. In this case, since

$$\log\, p(\mathbf{w}) = \frac{d}{2}\,\log\frac{\lambda}{2\pi} - \frac{\lambda}{2}\|\mathbf{w}\|_2^2,$$

we can equivalently formulate the optimization problem in (1) as

$$\max_{\mathbf{w},b}\ \Big\{F(\mathbf{w}, b|\mathcal{X}) - \frac{\lambda}{2}\|\mathbf{w}\|_2^2\Big\}. \tag{2}$$

(2) shows us that the problem reduces to an optimization problem with an ℓ_2-penalty.

3.2 The ℓ_1-Penalty Logistic Regression

In the second case, we impose a Laplace prior for $\mathbf{w}$, whose density is given by

$$\log\, p\,(\mathbf{w}) = d\,\log\frac{\gamma}{2} - \gamma\|\mathbf{w}\|_1.$$

With this prior, the optimization problem in (1) is equivalent to the following problem with the ℓ_1-penalty.

$$\max_{\mathbf{w},b}\ \big\{F(\mathbf{w}, b|\mathcal{X}) - \gamma\|\mathbf{w}\|_1\big\}. \tag{3}$$

The advantage of ℓ_1-penalty over ℓ_2-penalty is its utility in sparsity modeling [16] . Thus, ℓ_1-penalty logistic regression can serve for both classification and feature selection simultaneously.

4 Methodology

In this section, we first develop an approach to sublinear learning for ℓ_2-penalty logistic regression. We then extend the approach to ℓ_1-penalty case by adding certain conditions to achieve sparseness. Our approach is inspired by the *Elad-Tomer-Nathan* (ETN) framework in [11], a hybrid framework that deals with both hard margin and soft margin. Roughly speaking, our approach consists of three steps: deriving the hard margin and soft margin from the objective function, computing the derivative, and applying the ETN framework.

4.1 From ℓ_2-Penalty to Soft Margin

We treat the objective function (2) as two parts: likelihood and penalty. With this in mind, we introduce the notion of hard margin and soft margin to respectively represent these two parts.

In particular, we consider an alternative optimization problem under an ε-suboptimal solution basis. That is,

$$\max_{\mathbf{w},b,\xi_i\geq 0} \min_{i\in\{1,\cdots,n\}} f_i(\mathbf{w},b)+\xi_i \ \text{ s.t. } \|\mathbf{w}\|_2\leq 1 \ \text{ and } \ \sum_{i=1}^{n}\xi_i\leq n\nu. \tag{4}$$

In (4) $f_i(\mathbf{w},b)=\log g_i(y_i)$ is the hard margin part, while ξ_i is the soft margin part, and we have $\nu=-\frac{\sum_{i=1}^{n} f_i(\mathbf{w},b)}{n\|\mathbf{w}\|_2}$.

The following lemma shows the equivalent relationship between (2) and (4).

Lemma 1. *Let $(\mathbf{w}^\varepsilon,b^\varepsilon,\xi^\varepsilon)$ be an ε-suboptimal solution to the optimization problem (4) with optimal value κ, and consider the rescaled solution $\tilde{\mathbf{w}}=\mathbf{w}^\varepsilon/\kappa,\tilde{b}=b^\varepsilon/\kappa,\tilde{\xi}=\xi^\varepsilon/\kappa$. The the following two inequalities hold.*

$$\|\tilde{\mathbf{w}}\|_2\leq\frac{1}{1-\varepsilon\|\mathbf{w}\|_2}\|\mathbf{w}\|_2 \quad \textit{and} \quad F\left(\tilde{\mathbf{w}},\tilde{b}\right)\leq\frac{1}{1-\varepsilon\|\mathbf{w}\|_2}F(\mathbf{w},b).$$

The proof of Lemma 1 is given in Appendix A. Lemma 1 shows that solving (4) exactly yields Pareto optimal solutions of (2). Moreover, if we solve (4) via approximation, we obtain a suboptimal solution. As for parameters ν and ξ_i, we only need to consider $0\leq\nu\leq 1$ and $0\leq\xi_i\leq 2$.

4.2 Derivative of Objective Function

For hard margin, we compute the derivative of $f_i(\mathbf{w},b)$ with respect to $\mathbf{w}$. In this case, we have

$$f_i(\mathbf{w},b)=\log g_i(y_i). \tag{5}$$

The first partial derivative of (5) is as follows

$$\begin{aligned} coef &\triangleq \frac{\partial f_i(\mathbf{w},b)}{\partial\mathbf{w}}=-\frac{\partial\log[1+\exp(-y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i+b))]}{\partial\mathbf{w}} \\ &=\frac{\mathbf{x}_i y_i\exp(-y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i+b))}{1+\exp(-y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i+b))}=y_i g_i(-y_i)\mathbf{x}_i. \end{aligned} \tag{6}$$

In order to extend this result to ℓ_1-penalty logistic regression, we only need to adjust the derivative of $f_i(\mathbf{w},b)$ with respect to $\mathbf{w}$. In this case, we need to use the sub-differential of $\|\mathbf{w}\|_1$. First, we define a signum multi-function of $t\in\mathbb{R}$ as

$$S(t)\triangleq\partial|t|=\begin{cases}\{+1\} & \text{if } t>0\\ [-1,1] & \text{if } t=0\\ \{-1\} & \text{if } t<0.\end{cases}$$

For $\mathbf{x}\in\mathbb{R}^d$, we define $S(\mathbf{x})\in\mathbb{R}^d$ with $(S(\mathbf{x}))_i=S(x_i)$ for $i=1,\ldots,d$. Then the derivative of (3) is

$$coef=y_i g_i(-y_i)\mathbf{x}_i-\gamma S(\mathbf{w}). \tag{7}$$

Eqn. (7) is the simple and general form for $coef$.

4.3 The ETN Framework

The *Elad-Tomer-Nathan* framework [11] is a hybrid method to handle hard margin and soft margin separately and simultaneously. The ETN framework enjoys the property of fast convergence for both hard margin and soft margin.

Each iteration of the method works in two steps. The first one is the *stochastic primal update*:

(1) An instance $i \in \{1, \ldots, n\}$ is chosen according to a probability vector $\mathbf{p}$;
(2) The primal variable $\mathbf{w}$ is updated according to the derivative of $f_i(\mathbf{w}, b)$ and the soft margin, via an online update with regret.

The second one is the *stochastic dual update*:

(1) A stochastic estimate of $f_i(\mathbf{w}, b)$ plus the soft margin is obtained, which can be computed in $O(1)$ time per term;
(2) The probability vector $\mathbf{p}$ is updated based on the above computed terms by using the *Multiplicative Updates* (MW) framework [1] for online optimization over the simplex.

5 Algorithms and Analysis

We use the following notations in our algorithms and analysis.

$clip(\cdot)$ is a projection function defined as follows:

$$clip(a, b) \triangleq \max(\min(a, b), -b) \quad a, b \in \mathbb{R}.$$

$\operatorname{sgn}(\cdot)$ is the sign function; namely,

$$\operatorname{sgn}(x) = \begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0. \end{cases}$$

$g(\cdot)$ is the logistic function; namely,

$$g(x) = \frac{1}{1 + e^{-x}}$$

We let Λ be the $\mathbb{R}_n$ Euclidean space which meets the following conditions:

$$\Lambda = \{\xi \in \mathbb{R}_n \mid \forall i,\ 0 \le \xi_i \le 2,\ \|\xi\|_1 \le \nu n\}.$$

5.1 The Sublinear Algorithm for ℓ_2-Penalty Logistic Regression

We give the sublinear algorithm for ℓ_2-penalty logistic regression in Algorithm 1. In the pseudo-code of Algorithm 1, line 5 to line 11 is the primal part, where *coef* is the estimator of the derivatives and ξ is the soft margin. Line 12 to line

Algorithm 1. SLLR-L2

1: Input: $\varepsilon > 0, 0 \leq \nu \leq 1, X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^n$
2: Let $T \leftarrow 1000^2 \varepsilon^{-2} \log n, \eta \leftarrow \sqrt{\log(n)/T}$
3: $\mathbf{u}_0 \leftarrow \mathbf{0}_d, \mathbf{w}_1 \leftarrow \mathbf{0}_d, \mathbf{q}_1 \leftarrow \mathbf{1}_n, b_1 \leftarrow 0$
4: **for** $t = 1$ to T **do**
5: $\mathbf{p}_t \leftarrow \mathbf{q}_t / \|\mathbf{q}_t\|_1$
6: Choose $i_t \leftarrow i$ with probability $\mathbf{p}(i)$
7: Let $coef = y_{i_t} g\left(-y_{i_t}\left({\mathbf{w}_t}^T \mathbf{x}_{i_t} + b_t\right)\right)$
8: Let $\mathbf{u}_t \leftarrow \mathbf{u}_{t-1} + \frac{coef}{\sqrt{2T}} \mathbf{x}_{i_t}$
9: $\xi_t \leftarrow \text{argmax}_{\xi \in \Lambda}\left({\mathbf{p}_t}^T \xi\right)$
10: $b_t \leftarrow \text{sgn}\left({\mathbf{p}_t}^T \mathbf{y}\right)$
11: $\mathbf{w}_t \leftarrow \mathbf{u}_t / \max\{1, \|\mathbf{u}_t\|_2\}$
12: Choose $j_t \leftarrow j$ with probability $\mathbf{w}_t(j)^2 / {\|\mathbf{w}_t\|_2}^2$
13: **for** $i = 1$ to n **do**
14: $\sigma \leftarrow \mathbf{x}_i(j_t) {\|\mathbf{w}_t\|_2}^2 / \mathbf{w}_t(j_t) + \xi_t(i) + y_i b_t$
15: $\hat{\sigma} \leftarrow clip(\sigma, 1/\eta)$
16: $\mathbf{q}_{t+1}(i) \leftarrow \mathbf{q}_t(i)\left(1 - \eta\hat{\sigma} + \eta^2 \hat{\sigma}^2\right)$
17: **end for**
18: **end for**
19: Output: $\bar{\mathbf{w}} = \frac{1}{T}\sum_t \mathbf{w}_t, \bar{b} = \frac{1}{T}\sum_t b_t$

17 is the dual part, where σ serves as an estimator of $f_i(\mathbf{w}, b)$ plus the soft margin. σ also serves as the derivative of $\mathbf{p}(i)$. Although the computation of line 15 and 16 makes $\hat{\sigma}$ a biased approximation, it is critical to the stability of the algorithm. The resulting bias is negligible in our approach. This can be shown in the experimental results in [11]. Because of the similarity between our SLLR-L2 and SVM-SIMBA presented in [11], we can naturally invoke the statement here.

Note that the update of ξ_t in line 9 can be accomplished by using a simple greedy algorithm in $O(n)$ time. We can always set $\xi_t(i) = 2$ corresponding to the first $[\frac{\nu n}{2}]$ number of largest entries $\mathbf{p}(i)$ of $\mathbf{p}_t$ with respect to i. Then the residue $\nu n - 2[\frac{\nu n}{2}]$ is assigned to $\xi_t(\hat{i})$, where $\hat{i}$ is exactly the index of the $[\frac{\nu n}{2}]+1$ largest one in $\mathbf{p}_t$. Finally, we put $\xi_t(i) = 0$ elsewhere.

5.2 The Sublinear Algorithm for ℓ_1-Penalty Logistic Regression

In Algorithm 2, we give the sublinear approximation procedure for ℓ_1-penalty logistic regression. Here, we let $\mathbf{uprev}_t$ be $\mathbf{u}_{t-1}$ in the previous iteration. We achieve sparseness by adding pseudo-code from Line 11 to Line 16.

To make use of (7), we introduce some techniques to ensure the numerical convergence and stability. Considering the update computation in the primal step, we should make the following three rules.

(1) When $\mathbf{u}_t(j) = 0$, we do not apply $-\gamma S(\mathbf{w})$ and simply make the value 0 by default.

Algorithm 2. SLLR-L1

1: Input: $\varepsilon > 0, \gamma > 0, X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^n$
2: Let $T \leftarrow 1000^2 \varepsilon^{-2} \log n, \eta \leftarrow \sqrt{\log(n)/T}$
3: $\mathbf{u}_0 \leftarrow \mathbf{0}_d, \mathbf{wavg}_0 \leftarrow \mathbf{0}_d, \mathbf{q}_1 \leftarrow \mathbf{1}_n, b_1 \leftarrow 0$
4: **for** $t = 1$ to T **do**
5: $\mathbf{p}_t \leftarrow \mathbf{q}_t / \|\mathbf{q}_t\|_1$
6: $\mathbf{uprev}_t \leftarrow \mathbf{u}_{t-1}$
7: Choose $i_t \leftarrow i$ with probability $\mathbf{p}(i)$
8: Let $coef = y_{i_t} g\left(-y_{i_t}\left({\mathbf{wavg}_{t-1}}^T \mathbf{x}_{i_t} + b_t\right)\right)$
9: Let $\mathbf{u}_t \leftarrow \mathbf{u}_{t-1} + \frac{coef}{\sqrt{2T}} \mathbf{x}_{i_t}$
10: $b_t \leftarrow \text{sgn}\left({\mathbf{p}_t}^T \mathbf{y}\right)$
11: **for** $j = 1$ to d **do**
12: **if** $\mathbf{uprev}_t(j) > 0$ **and** $\mathbf{u}_t(j) > 0$
13: $\mathbf{u}_t(j) = \max(\mathbf{u}_t(j) - \gamma, 0)$
14: **if** $\mathbf{uprev}_t(j) < 0$ **and** $\mathbf{u}_t(j) < 0$
15: $\mathbf{u}_t(j) = \min(\mathbf{u}_t(j) + \gamma, 0)$
16: **end for**
17: $\mathbf{w}_t \leftarrow \mathbf{u}_t / \max\{1, \|\mathbf{u}_t\|_2\}$
18: $\mathbf{wavg}_t \leftarrow \frac{t-1}{t} \mathbf{wavg}_{t-1} + \frac{1}{t} \mathbf{w}_t$
19: Choose $j_t \leftarrow j$ with probability $\mathbf{w}_t(j)^2 / {\|\mathbf{w}_t\|_2}^2$
20: **for** $i = 1$ to n **do**
21: $\sigma \leftarrow \mathbf{x}_i(j_t) {\|\mathbf{w}_t\|_2}^2 / \mathbf{w}_t(j_t) + y_i b_t$
22: $\hat{\sigma} \leftarrow clip(\sigma, 1/\eta)$
23: $\mathbf{q}_{t+1}(i) \leftarrow \mathbf{q}_t(i)\left(1 - \eta\hat{\sigma} + \eta^2 \hat{\sigma}^2\right)$
24: **end for**
25: **end for**
26: Output: $\mathbf{wavg}_t, \bar{b} = \frac{1}{T} \sum_t b_t$

(2) In order to apply $-\gamma S(\mathbf{w})$ for sparseness, we set $\mathbf{u}_t(j) = 0$, if it changes between positive values and negative values after applying the derivative. This is showed in Line 13 and Line 15.

(3) To avoid a $\mathbf{0}$ vector when γ is large, we need to determine the derivative by a trend, not a single point. Thus, we consider two consecutive update steps of $\mathbf{u}_t(j)$. Line 12 and Line 14 ensure that if $\mathbf{u}_t(j)$ and $\mathbf{u}_{t-1}(j)$ are either both positive values or both negative ones, we apply the derivative, otherwise we do not change $\mathbf{u}_t(j)$. This is a logical approximation, and enables the small variance of values changing between positive values and negative ones.

For ℓ_1-penalty logistic regression, the derivative is much more sensitive with respect to $\mathbf{w}_t$, as it is sparse in the computation. So in line 8, when we compute $coef$, we change $\mathbf{w}_t$ to $\mathbf{wavg}_{t-1}$ in order to make our algorithm more computationally stable.

5.3 Running Time Analysis

We now formally describe the MW algorithm and give theorems for running times of our algorithms.

Definition 1. *(MW algorithm) [3]. Consider a sequence of vectors* $\mathbf{v}_1, ..., \mathbf{v}_T \in \mathbb{R}^d$ *and a parameter* $\eta > 0$*. The Multiplicative Weights (MW) algorithm is defined as follows: let* $\mathbf{w}_1 \leftarrow \mathbf{1}_n$*, and for* $t \geq 1$*,*

$$\mathbf{p}_t \leftarrow \mathbf{w}_t / \|\mathbf{w}_t\|_1, \quad \textit{and} \quad \mathbf{w}_{t+1}(i) \leftarrow \mathbf{w}_t(i) \left(1 - \eta \mathbf{v}_t(i) + \eta^2 \mathbf{v}_t(i)^2\right).$$

The following lemma establishes a regret bound for the MW algorithm.

Lemma 2. *(The Variance MW Lemma) [3]. The MW algorithm satisfies*

$$\sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{v}_t \leq \min_{i \in \{1,\ldots,n\}} \sum_{t=1}^{T} \max\{\mathbf{v}_t(i), -\frac{1}{\eta}\} + \frac{\log n}{\eta} + \eta \sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{v}_t^2$$

The following theorems give the running times of Algorithm 1 and Algorithm 2, respectively.

Theorem 1. *The SLLR-L2 algorithm returns an* ε*-approximate solution to the optimization problem of (4) with probability at least* $1/2$*. Its running time is* $\tilde{O}\left(\varepsilon^{-2}(n+d)\right)$.

We give the proof of Theorem 1 in Appendix B. Because the SLLR-L1 is essentially an extension of SLLR-L2, the running time is the same, and we omit the proof of Theorem 2 in this paper due to length constraint.

Theorem 2. *The SLLR-L1 algorithm returns an* ε*-approximate solution to the optimization problem of (3) with probability at least* $1/2$*. Its running time is* $\tilde{O}\left(\varepsilon^{-2}(n+d)\right)$

6 Experiments

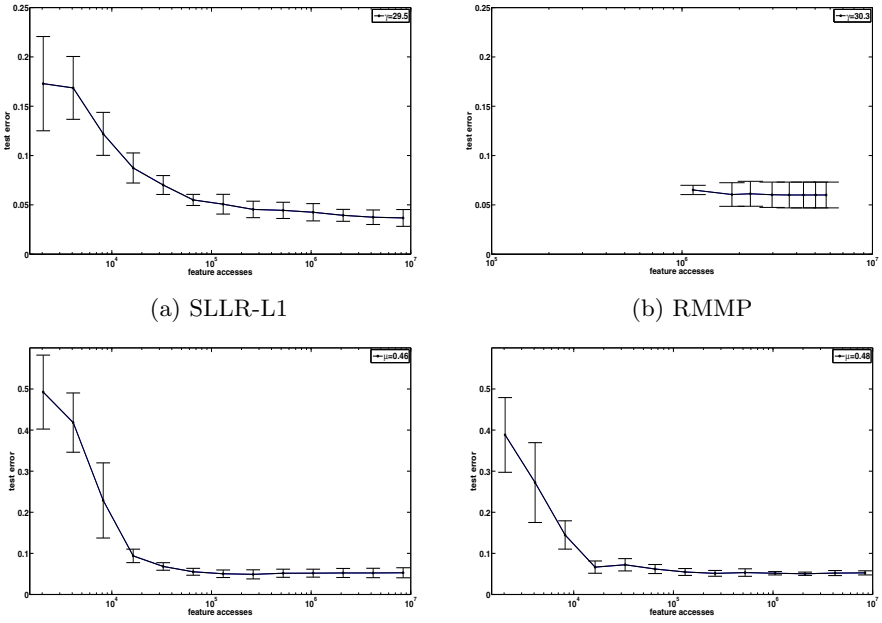
In this section, we conduct an empirical analysis of our algorithms. Particularly, we illustrate test errors in terms of feature accesses and convergence in terms of MAP. As illustrated in Section 1, feature accesses are the main cost in computation. They are good indicators of running time and best demonstrate the efficiency of the proposed algorithms. For SLLR-L2, we choose SVM-SIMBA algorithm [11] as a comparison baseline. For SLLR-L1, we choose the state-of-the-art RMMP [2] algorithm, a popular method for solving logistic regression with ℓ_1-penalty.

We choose three open datasets to run all four test programs: The **News-Group** dataset (after proper preprocessing) has 893 features and 1985 instances. We split it into a training set of 1390 instances and a test set of 595 instances. The second test dataset is the **Gisette** [8] dataset, which has 5000 features and 7000 instances. We split it into a training set of 6000 instances and a test set of 1000 instances. The third and final test dataset is the **ECUE Spam** [5] dataset, which has 197650 features and 10978 instances (after proper preprocessing). We split it into a training set of 9000 instances and a test set of 1978 instances. We randomly repeat such split 20 times and our analysis is based on the average performance of 20 repetitions.

6.1 Analysis of Performance

In all three experiments, we tuned parameters ν and γ of each algorithm based on the cross-validation method [13]. Note that our algorithms assume random access to features (as opposed to instances), thus it is not meaningful to compare the test error as a function of the number of iterations of each algorithm. Instead, according to our computational model, we compare the test error as a function of the number of feature accesses of each algorithm. The results, averaged over 20 repetitions, are presented in Figure 1, 2 and 3.

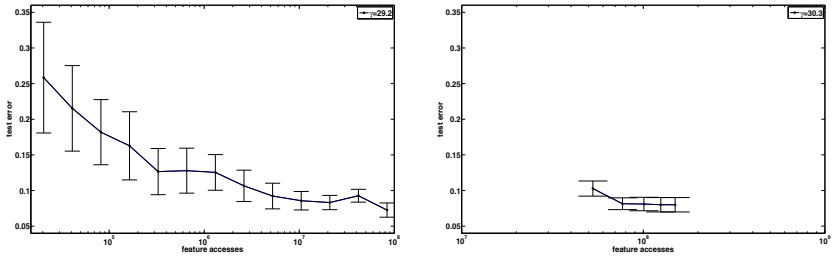
As can be seen from the figures, the performance of our SLLR-L2 algorithm is competitive with that of SIMBA on all the three datasets. With respect to ℓ_2-penalty, our experiments show that our SLLR-L2 algorithm can achieve a similar performance with SIMBA. With respect to ℓ_1-penalty, our SLLR-L1 algorithm can achieve a same level of performance as RMMP. Our SLLR-L1 algorithm has a fast convergence rate, which enables us to achieve an acceptable test error with much fewer feature accesses in comparison with RMMP (basically a batch algorithm).



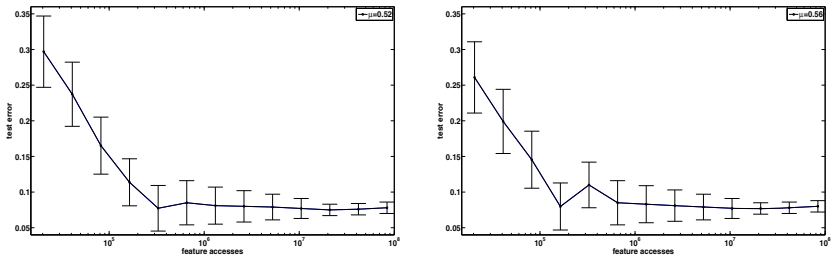
(a) SLLR-L1 (b) RMMP

(c) SLLR-L2 (d) SVM-SIMBA

Fig. 1. The test error, as a function of the number of feature accesses, on the **News-Group** dataset. For both SLLR-L1 and SLLR-L2, we set $\varepsilon = 0.5$.

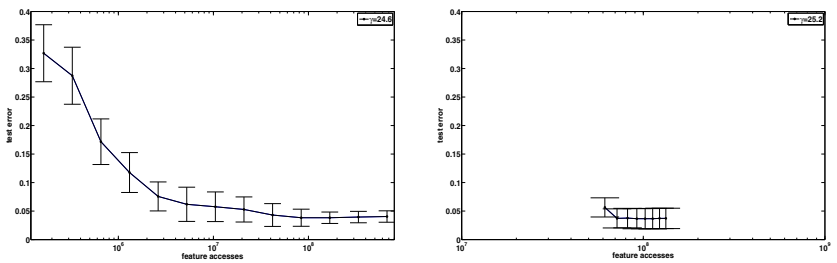


(a) SLLR-L1 (b) RMMP

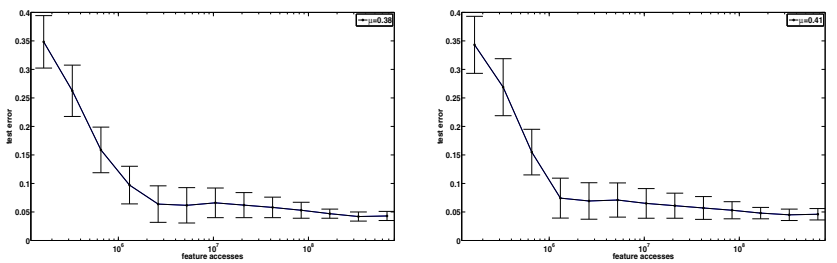


(c) SLLR-L2 (d) SVM-SIMBA

Fig. 2. The test error, as a function of the number of feature accesses, on the **Gisette** dataset. For both SLLR-L1 and SLLR-L2, we set $\varepsilon = 0.5$.



(a) SLLR-L1 (b) RMMP

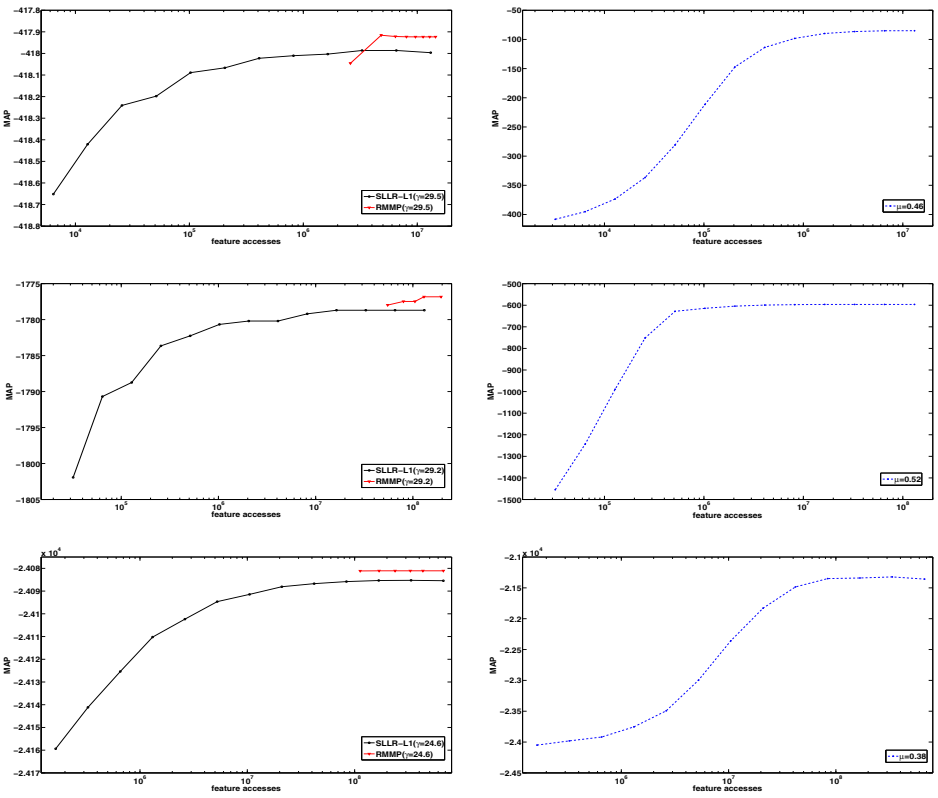


(c) SLLR-L2 (d) SVM-SIMBA

Fig. 3. The test error, as a function of the number of feature accesses, on the **ECUE Spam** dataset. For both SLLR-L1 and SLLR-L2, we set $\varepsilon = 0.5$.

6.2 Analysis of Convergence

Figures 4 shows the convergence of our algorithms. With respect to ℓ_2-penalty, we do not consider SVM-SIMBA, as its optimization objective function is not comparable with that of SLLR-L2. As the variance of MAP in different experiments is so small and does not contain much information, they are not shown in the figure for simplicity. The convergence of SLLR-L2 algorithm is very fast. There is a rapid growing of MAP, and it happens in a very early stage. With respect to ℓ_1-penalty, the optimum value achieved by our SLLR-L1 and RMMP, a state-of-art algorithm with a remarkable accuracy on MAP, is very close. Our SLLR-L1, though not strictly better than RMMP on accuracy, has a very small gap away from the optimum solution and it is acceptable considering the test error results shown previously. Moreover, our SLLR-L1 has the advantage of achieving its local optimum value much earlier than RMMP. This is because our



(a)SLLR-L1 and RMMP for ℓ_1-penalty (b)SLLR-L2 for ℓ_2-penalty

Fig. 4. The MAP, averaged over 20 random repetitions, as a function of the number of feature accesses, on the **NewsGroup**(first row), **Gisette**(second row), **ECUE Spam**(third row) datasets. For both SLLR-L1 and SLLR-L2, we set $\varepsilon = 0.5$.

approach loosely takes anywhere from 100 to 1000 times fewer feature accesses than RMMP.

7 Conclusion

In this paper, we have presented two efficient algorithms to solve PLR through the use of a stochastic approximation approach. In particular, we have devised two sublinear algorithms for the logistic regression models with ℓ_2-penalty and ℓ_1-penalty, respectively. Experimental results have illustrated that our algorithms work well on massive datasets and have significant computational performance over other existing methods for PLR. Our algorithms can also be easily applied to distributed storage systems with parallel update on all instances.

Appendix

A. Proof of Lemma 1

The method we use here is similar to that in [11].

Proof. We first consider the solution which is given by $\mathbf{w}^* = \mathbf{w}/\|\mathbf{w}\|_2, b^* = b/\|\mathbf{w}\|_2, \xi^* = \xi/\|\mathbf{w}\|_2$. So we have $\sum_{i=1}^n \xi_i^* = F(\mathbf{w}, b)/\|\mathbf{w}\|_2 = n\nu$. Then the optimal value is given by:

$$\kappa^* = \min_{i\in\{1,\cdots,n\}} \frac{f_i(\mathbf{w}, b) + \xi_i}{\|\mathbf{w}\|_2} = \frac{1}{\|\mathbf{w}\|_2}.$$

By the assumption on the suboptimality of $\mathbf{w}^\varepsilon, b^\varepsilon, \xi^\varepsilon$, we have $\kappa \geq \kappa^* - \varepsilon = \frac{1}{\|\mathbf{w}\|_2} - \varepsilon$, from which we can conclude that:

$$\|\tilde{\mathbf{w}}\|_2 = \frac{\|\mathbf{w}\|_2}{\kappa} \leq \frac{\|\mathbf{w}\|_2}{1/\|\mathbf{w}\|_2 - \varepsilon} \leq \frac{1}{1 - \varepsilon\|\mathbf{w}\|_2}\|\mathbf{w}\|_2.$$

From the form of the objective function, we also have:

$$F\left(\tilde{\mathbf{w}}, \tilde{b}\right) \leq \sum_{i=1}^n \tilde{\xi}_i \leq \frac{n\nu}{\kappa} \leq \frac{F(\mathbf{w}, b)}{\|\mathbf{w}\|_2} \cdot \frac{1}{1/\|\mathbf{w}\|_2 - \varepsilon} = \frac{F(\mathbf{w}, b)}{1 - \varepsilon\|\mathbf{w}\|_2}.$$

B. Proof of Theorem 1

The method we use here is similar to that in [11].
We first introduce some basic lemmas to simplify the proof.

Lemma 3. *For* $\sqrt{\log(n)/T} \leq \eta \leq 1/6$ *with probability at least* $1 - O(1/n)$*, it holds that*

$$\max_{i\in\{1,\cdots,n\}} \sum_{t=1}^T \mathbf{v}_t(i) - \sum_{t=1}^T \left[\mathbf{x}_i^T \mathbf{w}_t + \xi_t(i)\right] \leq 4\eta T,$$

$$\left|\sum_{t=1}^T \mathbf{p}_t^T \mathbf{v}_t - \sum_{t=1}^T \mathbf{p}_t^T (\mathbf{X}\mathbf{w}_t + \xi_t)\right| \leq 4\eta T.$$

Lemma 4. *For* $\sqrt{\log(n)/T} \leq \eta \leq 1/4$ *with probability at least* $1 - O(1/n)$*, it holds that*

$$\left|\sum_{t=1}^{T} \mathbf{x}_{i_t}^T \mathbf{w}_t - \sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{X} \mathbf{w}_t\right| \leq 12\eta T \text{ and } \left|\sum_{t=1}^{T} \mathbf{x}_{i_t}^T \mathbf{w}^* - \sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{X} \mathbf{w}^*\right| \leq 12\eta T.$$

Lemma 5. *With probability at least 3/4, it holds that*

$$\sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{v}_t^2 \leq 48T$$

We omit the proofs because they can be immediately obtained from [11] with some minor modifications.

Proof. Firstly, we prove the running time. Algorithm 1 makes $T = O(\varepsilon^{-2} \log n)$ iterations. Each iteration consists of two steps: the primal update and the dual update. The primal update contains a ℓ_1-sampling process for the choice of i_t ($O(n)$ time), and the update of $\mathbf{w}_t$ ($O(d)$ time). The update of ξ_t can be done using a simple greedy algorithm which takes $O(n)$ time. The primal update contains a ℓ_2-sampling process for the choice of j_t ($O(d)$ time), and an update of $\mathbf{p}$ ($O(n)$ time). Altogether, each iteration takes $O(n + d)$ time and the overall running time is therefore $\tilde{O}\left(\varepsilon^{-2}(n + d)\right)$.

Next, we analyze the output quality of Algorithm 1. Let γ^* be the value of the optimal solution of (4). Then, by the definition we have

$$\sum_{t=1}^{T} \mathbf{p}_t^T \left(\mathbf{X}\mathbf{w}^* + \xi^*\right) \geq T\gamma^*. \tag{8}$$

In the primal part of the algorithm we have

$$\sum_{t=1}^{T} \mathbf{x}_{i_t}^T \mathbf{w}_t \geq \sum_{t=1}^{T} \mathbf{x}_{i_t}^T \mathbf{w}^* - 2\sqrt{2T}.$$

Thus, from Lemma 4 we obtain that with probability $1 - O(1/n)$,

$$\sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{X} \mathbf{w}_t \geq \sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{X} \mathbf{w}^* - 2\sqrt{2T} - 24\eta T.$$

On the other hand,

$$\sum_{t=1}^{T} \mathbf{p}_t^T \xi_t \geq \sum_{t=1}^{T} \mathbf{p}_t^T \xi_t^*,$$

since ξ_t is the maximizer of $\mathbf{p}_t^T \xi$, and recalling (8), we get the following lower bound:

$$\sum_{t=1}^{T} \mathbf{p}_t^T \left(\mathbf{X}\mathbf{w}_t + \xi_t\right) \geq T\gamma^* - 2\sqrt{2T} - 24\eta T. \tag{9}$$

In the dual part of the algorithm, applying Lemma 2 on the clipped vector $\mathbf{v}_t$, we have that

$$\sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{v}_t \leq \min \sum_{t=1}^{T} \mathbf{v}_t + \frac{\log n}{\eta} + \eta \sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{v}_t^2,$$

and together with Lemma 3, we get that with probability $1 - O(1/n)$,

$$\sum_{t=1}^{T} \mathbf{p}_t^T \left(\mathbf{X}\mathbf{w}_t + \xi_t\right) \leq \min \sum_{t=1}^{T} \left(\mathbf{X}\mathbf{w}_t + \xi_t\right) + \frac{\log n}{\eta} + \eta \sum_{t=1}^{T} \mathbf{p}_t^T \mathbf{v}_t^2 + 8\eta T.$$

Hence, from Lemma 5, we obtain the following upper bound, with probability more than $\frac{3}{4} - O(1/n) \geq \frac{1}{2}$

$$\sum_{t=1}^{T} \mathbf{p}_t^T \left(\mathbf{X}\mathbf{w}_t + \xi_t\right) \leq \min \sum_{t=1}^{T} \left(\mathbf{X}\mathbf{w}_t + \xi_t\right) + \frac{\log n}{\eta} + 56\eta T. \tag{10}$$

Finally, combining bounds (9), (10) and dividing by T we have that with probability more than $\frac{1}{2}$,

$$\min \frac{1}{T} \sum_{t=1}^{T} \mathbf{p}_t^T \left(\mathbf{X}\mathbf{w}_t + \xi_t\right) \geq \gamma^* - \frac{2\sqrt{2}}{\sqrt{T}} - \frac{\log n}{\eta T} - 80\eta,$$

and using our choices for T and η, we conclude that with probability at least $\frac{1}{2}$, it holds that

$$\min \left(\mathbf{X}\mathbf{w}_t + \xi_t\right) \geq \gamma^* - \varepsilon.$$

This implies that the vectors $(\bar{\mathbf{w}}, \bar{\xi})$ form an ε-approximate solution.

References

1. Arora, S., Hazan, E., Kale, S.: The multiplicative weights update method: a meta algorithm and applications (2005), Preliminary draft of paper available online at http://www.cs.princeton.edu/~arora/pubs/MWsurvey.pdf (manuscript)
2. Balakrishnan, S., Madigan, D.: Algorithms for sparse linear classifiers in the massive data setting. The Journal of Machine Learning Research 9, 313–337 (2008)
3. Clarkson, K.L., Hazan, E., Woodruff, D.P.: Sublinear optimization for machine learning. In: Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, pp. 449–457. IEEE Computer Society (2010)
4. Cotter, A., Shalev-Shwartz, S., Srebro, N.: The kernelized stochastic batch perceptron. Arxiv preprint arXiv:1204.0566 (2012)
5. Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L.: A case-based technique for tracking concept drift in spam filtering. Knowledge-Based Systems 18(4-5), 187–195 (2005)
6. Garber, D., Hazan, E.: Approximating semidefinite programs in sublinear time. In: Advances in Neural Information Processing Systems (2011)
7. Genkin, A., Lewis, D.D., Madigan, D.: Large-scale bayesian logistic regression for text categorization. Technometrics 49(3), 291–304 (2007)

8. Guyon, I., Gunn, S., Ben-Hur, A., Dror, G.: Result analysis of the nips 2003 feature selection challenge. In: Advances in Neural Information Processing Systems, vol. 17, pp. 545–552 (2004)
9. Hastie, T., Tishirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2001)
10. Hazan, E., Koren, T.: Optimal algorithms for ridge and lasso regression with partially observed attributes. Arxiv preprint arXiv:1108.4559 (2011)
11. Hazan, E., Koren, T., Srebro, N.: Beating sgd: Learning svms in sublinear time. In: Advances in Neural Information Processing Systems (2011)
12. Hogan, C., Cassell, L., Foglesong, J., Kordas, J., Nemanic, M., Richmond, G.: The livermore distributed storage system: Requirements and overview. In: Tenth IEEE Symposium on Mass Storage Systems Digest of Papers, pp. 6–17. IEEE (1990)
13. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International Joint Conference on Artificial Intelligence, vol. 14, pp. 1137–1145. Lawrence Erlbaum Associates Ltd. (1995)
14. Panda, D.K.: Global reduction in wormhole k-ary n-cube networks with multidestination exchange worms. In: IPPS: 9th International Parallel Processing Symposium, pp. 652–659. IEEE Computer Society Press (1995)
15. Shi, J., Yin, W., Osher, S., Sajda, P.: A fast hybrid algorithm for large scale l1-regularized logistic regression. Journal of Machine Learning Research 1, 8888 (2008)
16. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267–288 (1996)
17. Tsumoto, S.: Mining diagnostic rules from clinical databases using rough sets and medical diagnostic model. Information Sciences 162(2), 65–80 (2004)
18. Vapnik, V.: Statistical Learning Theory. John Wiley and Sons, New York (1998)
19. Xiao, L.: Dual averaging methods for regularized stochastic learning and online optimization. The Journal of Machine Learning Research 11, 2543–2596 (2010)
20. Zhang, T.: Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 116. ACM (2004)

Author Name Disambiguation Using a New Categorical Distribution Similarity

Shaohua Li, Gao Cong, and Chunyan Miao

Nanyang Technological University
shaohua@gmail.com, {gaocong,ascymiao}@ntu.edu.sg

Abstract. Author name ambiguity has been a long-standing problem which impairs the accuracy of publication retrieval and bibliometric methods. Most of the existing disambiguation methods are built on similarity measures, e.g., "Jaccard Coefficient", between two sets of papers to be disambiguated, each set represented by a set of categorical features, e.g., coauthors and published venues[1]. Such measures perform bad when the two sets are small, which is typical in Author Name Disambiguation. In this paper, we propose a novel categorical set similarity measure. We model an author's preference, e.g., to venues, using a categorical distribution, and derive a likelihood ratio to estimate the likelihood that the two sets are drawn from the same distribution. This likelihood ratio is used as the similarity measure to decide whether two sets belong to the same author. This measure is mathematically principled and verified to perform well even when the cardinalities of the two compared sets are small. Additionally, we propose a new method to estimate the number of distinct authors for a given name based on the name statistics extracted from a digital library. Experiment shows that our method significantly outperforms a baseline method, a widely used benchmark method, and a real system.

Keywords: Name Disambiguation, Categorical Sampling Likelihood Ratio.

1 Introduction

Bibliometrics is an important methodology to assess the output and impact of researchers and institutions. Ambiguous names which correspond to many authors are a long-standing headache for bibliometric assessors and users of digital libraries. For example, in DBLP, there are at least 8 authors named *Rakesh Kumar*, and their publications are mixed in the retrieved citations. The ambiguity on Chinese names is more severe, as many Chinese share a few family names such as *Wang*, *Li*, and *Zhang*. An extreme example is *Wei Wang*. According to our labeling, it corresponds to over 200 authors in DBLP! As more and more researchers become active, the ambiguity problem will only become graver.

[1] Venues here refer to the journal or conference, such as *J. ACM* or *SIGIR*.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 569–584, 2012.

Author Name Disambiguation refers to splitting the bibliographic records by different authors with the same name into different clusters, so that each cluster belongs to one author and each author's works are gathered in one cluster.

For each paper, we consider 3 features: *coauthors*, *published venue* and *title*, by following the setting used in previous work [5,3,12]. Under this setting, our proposed method can be general and applicable to the existing bibliography databases, e.g., DBLP, since they contain information on the three features for each paper. Each feature serves as a body of evidence used to decide whether two homonymous authors are the same person. *Coauthors* and *venues* are two important features that have categorical values. During disambiguation, we need measure the similarity between two clusters of papers. Naturally the feature values in each cluster form a set of categorical data, and thus a categorical set similarity measure is an important foundation of a disambiguation algorithm.

Given two sets of categorical data, previous methods of name disambiguation use set similarity measures, such as *Jaccard Coefficient* ([2,12]) or *cosine similarity* ([8]), which often fail when the sets are unbalanced in cardinality, or when the frequencies of the elements in each set have distinctive patterns (to be explained in Section 4). We exploit the property that categorical sets from the same author follow similar distributions, and propose a generative probabilistic model to estimate the similarity of two sets. We name this novel similarity measure as **Categorical Sampling Likelihood Ratio (CSLR)**.

In addition, the ambiguity (number of distinct people) of a disambiguated name needs to be estimated to guide the disambiguation process. We exploit the property that the different parts of a person name in a given culture are chosen roughly independently, and derive a simple statistical method to estimate the ambiguity, based only on the name statistics in a digital library. The estimated ambiguity is shown to be reasonably close to the actual value for Chinese names.

We evaluate our system on two test sets extracted from the January 2011 dump of DBLP. Experiments show that our method significantly outperform one baseline method (by 2-12%), a representative previous method DISTINCT (by 4-13%) and a well-known system Arnetminer [9] (`http://arnetminer.org/`) (by 6-17%) in terms of macro-average F1 scores.

The rest of this paper is organized as follows. In Section 2, we review related work. In Section 3, we define basic notations used in this paper, and state the objective of Author Name Disambiguation. In Section 4, we establish the novel set similarity measure CSLR. In Section 5, we outline our clustering system based on CSLR. In Section 6, we describe the *name ambiguity estimation* method. In Section 7, we report experimental results. Finally, we conclude in Section 8. In addition, all proofs are in the full version of this paper ([6]). The source code and data set are available at `http://github.com/askerlee/namedis`.

2 Related Work

A pioneering work [5] on Author Name Disambiguation presents two supervised learning approaches, using Naive Bayes and SVM, respectively. For each name to

be disambiguated, a specific classifier is trained. Therefore, hand-labeled papers for each name are needed. This overhead is unaffordable in practice.

The method DISTINCT [12] uses SVM to learn the weights of features. The training data for SVM is generated automatically. The title is considered the unigram "bag-of-words" (BoW). Each cluster of papers has a few features, and the similarity between feature value sets of two clusters is calculated using Jaccard Coefficient. As another similarity measure, the connection strength between clusters is measured by a random walk probability. The two similarity measures are combined and form the similarity used in the agglomerative clustering.

The work [2] formulats the Name Disambiguation problem as a hypergraph, where each author is one node. Relationships among authors, such as the coauthorship of a few authors, are represented as hyperedges. The similarity between two clusters is measured by comparing their "neighboring sets" (other clusters they connect with), using Jaccard Coefficient or Adamic/Adar Similarity.

Torvik et al. ([10]) develops a disambiguation system on MEDLINE. First a training set is automatically generated, and the likelihood ratio of each feature value as its evidential strength is estimated from the training set. Evidence provided by different feature values is aggregated under the Naive Bayes assumption, and the probability that two papers belong to the same author is estimated. Finally, a maximum likelihood agglomerative clustering is conducted.

Recently, Tang et al. ([8,11]) presents two closely-related methods based on Pairwise Factor Graph models. The authorship is modeled as edges between observation variables (papers) and hidden variables (author labels). Features of each paper, and relationships such as CoPubVenue and CoAuthor, have impact on the probability of each assignment of labels. The similarity between two clusters is encoded in different "factors" (edge potentials) on different features. The clustering process tries different author label assignments and finds the one with maximal probability. Moreover, [11] improves the disambiguation resultes based on user feedback, and is being used online in Arnetminer for disambiguation (`http://arnetminer.org/disambiguation`).

In addition to the title, co-authorship and venue information, authors' homepages ([11]), and results returned by a search engine ([7]) are also used for disambiguation. However, such information is not always available.

3 Problem Formulation

In a digital library, each author name e may correspond to one or more authors $\{a_1, a_2, \cdots, a_{\kappa(e)}\}$. Each a_i is called e's **namesake**. The number of namesakes $\kappa(e)$ is the **ambiguity** of name e. The estimated ambiguity is denoted by $\hat{\kappa}(e)$. The name e being disambiguated is called the **focus name**. Each paper d has a set of authors $A_d = \{a_1, a_2, \cdots, a_m\}$. Suppose a_i has name e. The rest authors (if any) $A_d \setminus \{a_i\}$ are the **coauthors** with regard to paper d, denoted by $\mathrm{co}(d)$.

We represent a collection of categorical data as a **multiset**. In contrast to the traditional set, here each element x in set S has a frequency value $\mathrm{freq}_S(x)$. $\mathrm{freq}_S(x)$ could be a real number after scaling. The **cardinality** of a multiset S,

denoted by $|S|$, is the sum of frequencies of all its elements: $|S| = \sum_{x \in S} \text{freq}_S(x)$. A multiset S is often represented as a list of pairs as $\{x_1 : f_1, \cdots, x_m : f_m\}$, where $f_i = \text{freq}_S(x_i)$. Often we simply refer to a multiset as a **set** when the meaning is clear from context.

Given a set of papers $C = \{d_1, d_2, \cdots, d_n\}$ written by author a, the **coauthor set** of C is the union of coauthors[2] of all d_i, i.e., $\text{co}(C) = \cup_{i=1}^{n} \text{co}(d_i)$. Each coauthor $b_i \in \text{co}(C)$ has a frequency $\text{freq}_{\text{co}(C)}(b_i)$, which is the count of papers in C having b_i as a coauthor.

Likewise, we refer to the multiset of publication venues for the set of papers C as the **venue set** of C, denoted by $V(C)$. Each venue $v_i \in V(C)$ has a frequency $\text{freq}_{V(C)}(v_i)$, which is the number of papers in C published in v_i.

Problem Statement. Given a focus name e and a set of papers authored by name e: $\mathcal{P}(e) = \{d_1, d_2, \cdots, d_n\}$, the problem of **name disambiguation** is to partition $\mathcal{P}(e)$ into different clusters $\{C_1, \cdots, C_{\kappa(e)}\}$, so that all papers in C_i are authored by person a_i and all the papers in $\mathcal{P}(e)$ by a_i are in C_i.

Before we present the proposed method for name disambiguation in Section 5, we first present the proposed similarity measure in Section 4, which lays the foundation of our method.

Table 1. Notation table

Notation	Description
e	An ambiguous name
$\kappa(e)$	Ambiguity of name e
a_i	An author (with no ambiguity)
C	A cluster of papers that belong to the same author
$\text{co}(C)$	Coauthor multiset of C: the union of coauthors of all papers in C
$V(C)$	Venue multiset of C: the union of venues of all papers in C
$\text{freq}_S(x)$	Frequency of an element x in a multiset S
S	A multiset, where each element $x \in S$ has a frequency
$\|S\|$	Cardinality of a multiset, i.e., the sum of frequencies of all elements
$\boldsymbol{p} = (p_0, p_1, \cdots, p_m)$	A parameter vector of a categorical distribution
B	Base Set (the larger one of two compared multisets S_1 and S_2)
BCD, $\mathcal{B}$	Base Categorical Distribution where B is drawn
A	Sampled Set (the smaller one of two multisets S_1 and S_2)
$\tilde{A}$	Conflated sampled set (all "unseen" outcomes become UNSEEN)
A'	Tolerated sampled set (by reducing some UNSEEN counts from $\tilde{A}$)
$\text{Cat}(\boldsymbol{p})$	A categorical distribution with the parameter vector $\boldsymbol{p}$
$\Pr(S\|\boldsymbol{p})$	Probability of drawing set S from $\text{Cat}(\boldsymbol{p})$
$S \sim \mathcal{D}$	The case of drawing S from distribution $\mathcal{D}$
$\Lambda(A, B)$	Categorical Sampling Likelihood Ratio (CSLR) between A and B

[2] As different coauthors with the same name are literally indistinguishable, the *coauthor* here may correspond to more than one actual author.

4 Categorical Sampling Likelihood Ratio – A Categorical Set Similarity Measure

In Section 4.1, we use a categorical distribution to model the preference of each author, introduce the intuition behind Categorical Sampling Likelihood Ratio (CSLR), and formulate CSLR as the ratio of two likelihoods. In Section 4.2, we present methods to approximate the two likelihoods. Section 4.3 presents the proposed CSLR.

For ease of discussion, we present CSLR in the context of two venue sets, each representing a set of papers by an author. The comparison between two coauthor sets can be computed similarly.

4.1 Modeling Using the Categorical Distribution and Motivation

Each author has preferences to the publication venue, and such preferences can be represented as a categorical distribution, namely the *Preference Distribution*. The frequency that the author published in a venue reflects the preference of this author to the venue. Consider a cluster of papers C belonging to author a. The venue of each paper in C is an observation of the preference distribution, and the whole venue set $V(C)$ forms a **sample** of that distribution. Suppose there are m possible outcomes (i.e., venues) in this distribution, denoted by $x_i,\ i = 1, \cdots, m$. Each x_i has a probability p_i drawn from this distributuion. We denote all the outcome probabilities as a vector: $\boldsymbol{p} = (p_1, \cdots, p_m)$. A categorical distribution with a parameter vector $\boldsymbol{p}$ is denoted by $\mathrm{Cat}(\boldsymbol{p})$. Therefore author a's preference distribution is $\mathrm{Cat}(\boldsymbol{p})$.

Different authors usually have distinctive preference distributions. Hence we can estimate the possibility that two clusters belong to the same author, by comparing the two distributions from which these venue sets are drawn. Such a problem is traditionally known as the *two-sample problem* ([4]).

The biggest challenge of the two-sample problem in Author Name Disambiguation is: during the clustering, a cluster of papers are often a small fragment of the complete set of papers by that author, and therefore the venue set is a **small sample** and often only a **partial observation** of the preference distribution. It is difficult to compare two distributions based only on two partial observations. Traditional categorical set/distribution similarity measures, such as *Jaccard Coefficient*: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$, its variant *Adamic/Adar Similarity*, *cosine similarity*, or *Kullback-Leibler divergence*, perform well when the sets A and B are large and good approximations of the underlying distributions, but do not fit well with Author Name Disambiguation. We take Jaccard Coefficient to illustrate the problems of these measures:

1. Sets A and B often have unbalanced cardinalities, and $J(A, B)$ is sensitive to the relative set cardinalities. In the extreme case that $A \subset B$, intuitively A, B are probably drawn from the same distribution (A is a smaller sample); however $J(A, B) = \frac{|A|}{|B|}$ varies drastically with the cardinality of either set;

2. The evidential strength of each shared element is usually regarded as the same, regardless of their relative importance. But some elements are more discriminative than others. For example, suppose x is the most frequent elemnt in B, but absent in A. Then it is strong evidence that A and B follow different distributions, and are dissimilar. But if x appears once in B and absent in A, it is only weak evidence. Note adding weights to elements does not help much, e.g., *Adamic/Adar Similarity*, the weighted version of Jaccard Coefficient, is shown to perform worse than Jaccard Coefficient ([2]).

To this end, we propose a new measure. Assume two multisets A, B have arisen under one of the two hypotheses H_0 and H_1. The null hypothesis H_0 here is: A and B are drawn from different distributions (and thus belong to different authors). The alternative hypothesis H_1 is: A and B are drawn from the same distribution (and thus belong to the same author). We want to see how likely one hypothesis holds relative to the other. The more likely H_1 is relative to H_0, the more similar are A and B.

Formally, we estimate both $\Pr(H_1|B,\ A)$ and $\Pr(H_0|B,\ A)$. We compare these two posterior probabilities and get a likelihood ratio $\Lambda = \dfrac{\Pr(H_1|B,\ A)}{\Pr(H_0|B,\ A)}$. We use the likelihood ratio as the similarity between A and B.

We assume a flat prior on the two hypotheses: $\Pr(H_0) = \Pr(H_1) = 0.5$. By applying Bayes' theorem (the proof can be found in [6]), we get

Theorem 1

$$\Lambda = \frac{\Pr(H_1|B,\ A)}{\Pr(H_0|B,\ A)} = \frac{\Pr(A|B,\ H_1)}{\Pr(A|B,\ H_0)}.$$

To compute the likelihood ratio, we need to compute the two probabilities that A is seen, given B and one of the hypotheses, H_0 and H_1.

4.2 Calculating the Two Likelihoods

Computing $\Pr(A|B,\ H_1)$. Consider two authors a_1 and a_2, whose preference distributions are $\mathrm{Cat}(\boldsymbol{p_1})$ and $\mathrm{Cat}(\boldsymbol{p_2})$, respectively, and whose venue sets are A and B, respectively.

We proceed to estimate $\Pr(A|B,\ H_1)$. First, suppose hypothesis H_1 holds. Then $\boldsymbol{p_1} = \boldsymbol{p_2}$. This implies, given B and H_1, A is drawn from $\mathrm{Cat}(\boldsymbol{p_2})$. Let $\Pr(A|\boldsymbol{p_2})$ be the probability that A is drawn from $\mathrm{Cat}(\boldsymbol{p_2})$. Then $\Pr(A|B,\ H_1) = \Pr(A|\boldsymbol{p_2})$.

We estimate $\boldsymbol{p_1}, \boldsymbol{p_2}$ from A and B and get $\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2$, respectively. Then

$$\Pr(A|B,\ H_1) = \Pr(A|\boldsymbol{p_2}) \approx \Pr(A|\hat{\boldsymbol{p}}_2).$$

Note in Theorem 1, A and B are symmetric and exchangeable. Empiracally a larger sample tends to better reflect the actual distribution $\mathrm{Cat}(\boldsymbol{p_i})$. Without loss of generality, suppose $|B| \geq |A|$. Then $\mathrm{Cat}(\hat{\boldsymbol{p}}_2)$ is probably a better estimation of $\mathrm{Cat}(\boldsymbol{p}_2)$ than $\mathrm{Cat}(\hat{\boldsymbol{p}}_1)$ as an estimation of $\mathrm{Cat}(\boldsymbol{p}_1)$. The likelihood $\Pr(A|\hat{\boldsymbol{p}}_2)$ would likely be more accurate than $\Pr(B|\hat{\boldsymbol{p}}_1)$. So we choose B as the conditioning set, namely the *Base Set*, from which we estimate a *Base Categorical Distribution*

(BCD) $\mathcal{B}$, and the smaller set A as the conditioned set, namely the *Sampled Set*. If $|A| > |B|$, we simply exchange A and B.

Let us denote the base set as $B = \{x_1 : f_1,\ x_2 : f_2,\ \cdots,\ x_n : f_n\}$, and the sampled set as $A = \{y_1 : g_1,\ y_2 : g_2,\ \cdots,\ y_m : g_m\}$, where x_i, y_j are outcomes (venues), and $f_i = \text{freq}_B(x_i)$, $g_j = \text{freq}_A(y_j)$. We can estimate $\mathcal{B}$ from B using Maximum Likelihood Estimation (MLE): $\hat{p}_i = \frac{f_i}{\sum_i f_i}$.

Considering that B may not cover all the outcomes in $\mathcal{B}$, we should tolerate outcomes in A but not in B. We introduce a "wildcard" outcome: UNSEEN (denoted by x_0, drawn with a small probability p_0). Any outcome in A but not in B is treated as UNSEEN, without discrimination. We adopt the widely used *Jeffreys prior*([1]) to assign a pseudocount $\delta = 0.5$ to UNSEEN and all the observed outcomes in B. The smoothed estimator gives the following parameters:

$$\hat{p}_0 = \frac{\delta}{\delta(n+1) + \sum_i f_i}, \qquad \hat{p}_i = \frac{f_i + \delta}{\delta(n+1) + \sum_i f_i},\ \text{for } i = 1, \cdots, n. \tag{1}$$

The estimated $\mathcal{B}$ is $\hat{\mathcal{B}} = \text{Cat}(\hat{\boldsymbol{p}}_2) = \text{Cat}(\hat{p}_0, \hat{p}_1, \cdots, \hat{p}_n)$.

Before calculating the probability that A is drawn from $\hat{\mathcal{B}}$, we partition A into two sets – the "seen" outcomes A_s and the "unseen" ones A_u, and conflate A_u into UNSEEN:

1) $A_s = A \cap B$. Suppose $A_s = \{y_1 : g_1,\ \ldots,\ y_t : g_t\}$. We align (relabel) the elements in B with A_s, so that $x_i = y_i$, for $i = 1, \ldots, t$ (the remaining outcomes in B are labeled as $x_{t+1}, \cdots, x_n$ arbitrarily). Then outcome y_i is drawn with probability $\hat{p}_i$ from $\hat{\mathcal{B}}$;
2) $A_u = A \setminus B$ is the unseen outcomes. Suppose $A_u = \{y_{t+1} : g_{t+1},\ \ldots,\ y_m : g_m\}$. All elements in A_u are "conflated" to UNSEEN (x_0). Let the frequency of x_0 be g_0, then $g_0 = |A_u| = \sum_{i=t+1}^{m} g_i$.

We denote the conflated set as $\tilde{A}$. We have $\tilde{A} = \{x_0 : g_0,\ y_1 : g_1,\ \ldots,\ y_t : g_t\}$. Note the conflation does not change the cardinality of the set, i.e., $|\tilde{A}| = |A|$. Then the probability that drawing A from distribution $\mathcal{B}$, denoted by $A \sim \mathcal{B}$, is approximated by the probability that $\tilde{A} \sim \hat{\mathcal{B}}$:

$$\Pr(A|B,\ H_1) \approx \Pr(\tilde{A}|\hat{\boldsymbol{p}}_2) = \binom{|A|}{g_0,\ g_1, \cdots, g_t} \hat{p}_0^{g_0} \prod_{i=1}^{t} \hat{p}_i^{g_i}, \tag{2}$$

where $\binom{|A|}{g_0, g_1, \cdots, g_t}$ is the multinomial coefficient, counting the total number of sequences with the same frequencies of outcomes as in A.

Toleration of Preference Divergence: Converting from A to A'. The preference distribution of an author often evolves slowly with time. Thus an author has different preference distributions at different periods; however typically these categorical distributions share many common outcomes, and the probabilities of shared outcomes are still close. Thus the difference between the preference distributions of the same author at different times is usually much smaller than the difference between the distributions of different authors.

Consider two sets A and B, both belonging to author a, are drawn from slightly different preference distributions $\text{Cat}(\boldsymbol{p_1})$ and $\text{Cat}(\boldsymbol{p_2})$, respectively,

where the parameter vectors $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ are similar but not identical. Let B be the base set, and $\hat{\mathcal{B}}$ is the estimated BCD. When we calculate the probability that $A \sim \hat{\mathcal{B}}$, A may contain a few "unseen" outcome occurrences with respect to $\hat{\mathcal{B}}$, as well as a lot of "seen" outcome occurrences. These UNSEEN occurrences are all assigned a tiny probability $\hat{p}_0$, and contribute $c \cdot \hat{p}_0^{g_0}$ (c is a small factor in the multinomial coefficient) in (2), which reduces the probability drastically (although the majority of outcome occurrences are "seen"), wrongly indicating that A and B unlikely belong to the same author. The "culprit" of this undesirable result is the few "unseen" outcomes. In other words, the direct likelihood estimation is too stringent and intolerant to deviation from $\hat{\mathcal{B}}$.

To allow for preference divergence, before we calculate the likelihood, we reduce some count of UNSEEN, proportional to the cardinality of A. This strategy is called *toleration*. The kept outcome occurrences form a new *Tolerated Set* A'.

To perform toleration on set A, first we conflate the "unseen" outcomes in A and get $\tilde{A}$. Parameter θ_t controls the UNSEEN count to be reduced relative to A's cardinality, i.e., UNSEEN frequency g_0 will be reduced by $\theta_t|A|$. If UNSEEN frequency $g_0 < \theta_t|A|$, then the new frequency $g_0' = 0$. We set $\theta_t = \frac{1}{3}$. We denote the tolerated set as $A' = \{x_0 : h_0, y_1 : h_1, \cdots, y_r : h_r\}$, where $h_0 = g_0'$, and $h_i = \text{freq}_A(y_i)$, for $\forall i > 0$. The probability in (2) becomes $\Pr(A'|\hat{\boldsymbol{p}}_2)$:

$$\Pr(A|B, H_1) \approx \Pr(A'|\hat{\boldsymbol{p}}_2) = \binom{|A'|}{h_0, h_1, \cdots, h_r} \hat{p}_0^{h_0} \prod_{i=1}^{r} \hat{p}_i^{h_i}. \tag{3}$$

Computing $\Pr(A'|B, H_0)$. In the following, the sampled set in our likelihood estimation is the tolerated set A'. We will estimate $\Pr(A'|B, H_0)$ first.

The hypothesis H_0 states that A' and B are drawn from different categorical distributions, i.e., A' is drawn from a distribution other than $\text{Cat}(\boldsymbol{p}_2)$. Since any randomly-chosen categorical distribution is probably dissimilar to $\text{Cat}(\boldsymbol{p}_2)$, we can approximate $\Pr(A'|B, H_0)$ by $\Pr(A')$, i.e., the probability that A' is drawn from a categorical distribution $\text{Cat}(\boldsymbol{p})$, where we have no information about $\boldsymbol{p}$.

We limit the sample space of any possible categorical distribution $\text{Cat}(\boldsymbol{p})$ to the set of outcomes in $\mathcal{B}$: $\{x_1, \cdots, x_n\}$. Naturally, we assume a flat Dirichlet $\text{Dir}(\mathbf{1}_n)$ as the prior distribution of $\boldsymbol{p}$, where $\mathbf{1}_n = (1, \cdots, 1)$ is n dimensional.

Suppose $A' = \{x_0 : h_0, y_1 : h_1, \cdots, y_{r-1} : h_{r-1}, y_r : h_r\}$, then we can represent A' by the frequency vector of its elements: $\boldsymbol{h} = (h_0, h_1, \cdots, h_r, h_{r+1}, \cdots, h_n)$, where $h_{r+1} = \cdots = h_n = 0$. Then we have the following Theorem.

Theorem 2

$$\Pr(A'|B, H_0) \approx \Pr(A') = \int_{\boldsymbol{p}} \Pr(\boldsymbol{h}|\boldsymbol{p}) \Pr(\boldsymbol{p}; \mathbf{1}_n) d\boldsymbol{p} = \frac{1}{\binom{|A'|+n}{n}}, \tag{4}$$

where $\Pr(\boldsymbol{p}; \mathbf{1}_n)$ *denotes the probability of drawing* $\boldsymbol{p}$ *from* $\text{Dir}(\mathbf{1}_n)$.

The proof can be found in [6]. Theorem 2 reveals an interesting fact: $\Pr(A')$ is only determined by $|A'|$, A's cardinality, and n, the number of categories in B, but irrelevant to the histogram of outcome frequencies in A'.

4.3 Categorical Sampling Likelihood Ratio (CSLR)

As we have obtained two approximations of the two likelihoods in Eq. (3) and Theorem 2, we combine them and get the approximation of Λ:

$$\Lambda \approx \frac{\Pr(A'|\hat{\boldsymbol{p}}_2)}{\Pr(A')} = \binom{|A'|}{h_0,\, h_1,\, \cdots,\, h_r} \binom{|A'|+n}{n} \hat{p}_0^{h_0} \prod_{i=1}^{r} \hat{p}_i^{h_i}. \tag{5}$$

We name Λ as *Categorical Sampling Likelihood Ratio* (CSLR). It is directly used as the similarity between two categorical sets, such as venue sets and coauthor sets. For two sets A and B, we denote their CSLR as $\Lambda(A, B)$.

5 Clustering Framework

5.1 Overview of the Clustering Procedure

We use Agglomerative Clustering as the basic framework. It starts with each paper being a cluster, and at each step we find the most similar (the similarity measures will be defined later) pairs of clusters, and merge them, until the maximal similarity falls below certain threshold, or the cluster number is smaller than the estimated ambiguity of the disambiguated name. The whole clustering process divides into two stages:

1. Merge based on the evidence from shared coauthors;
2. Merge based on the combined similarity defined on the title sets and venue sets of each pair of clusters.

The reasons for developing the two-stage clustering are twofold: First, coauthors generally provide stronger evidence than other features, based on which the generated cluster usually comprises of papers of the same author, but the papers of an author may distribute among multiple clusters ([3]); Second, the venue and title features are relatively weak evidence, based on which we can further merge clusters from the same author.

5.2 Stage 1: Merging by Shared Coauthors

The existing work ([5,12,10,2,3]) usually takes shared coauthors as a crucial feature. They usually treat all authors equally, and combine two clusters if they have shared coauthors. However, we observe that the strength of the evidence provided by a shared coauthor varies from one to another. If a coauthor collaborates with many people, it is likely that the coauthor collaborate with different people with the same focus name. Especially when the focus name to be disambiguated has high ambiguity, the chance of different people sharing the same coauthor names would be high. Hence, we propose to distinguish those weak evidential coauthors from the strong evidential coauthors and treat them differently. For example, consider to disambiguate "Wei Wang". Coauthors *Jiawei Han* and *Jian Pei* both collaborate with different "Wei Wang". We observe that both *Jiawei Han* and *Jian Pei* have over 200 collaborators, and thus they should be treated as weak evidential coauthors when disambiguating "Wei Wang".

We proceed to present a statistical approach to estimating the probability that a coauthor b works with only one namesake of a given name e. Given that a coauthor b is shared by two clusters C_1 and C_2, the alternative hypothesis H_1 says C_1 and C_2 belong to the same author. If $\Pr(H_1|b)$ is large enough ($\geq \theta_{co}$), then b is regarded as *strong evidential*, and we merge C_1 and C_2. Otherwise b is *weak evidential*. Here θ_{co} is the decision threshold. We choose $\theta_{co} = 0.95$.

Let e be the disambiguated focus name. Suppose that the coauthor b randomly chooses n authors from the whole author set $\mathbb{A}$[3] to collaborate with, and among the n collaborators at least one person a_1 has name e. The total count of authors is denoted by $M = |\mathbb{A}|$. We assume the choice of collaboration follows a uniform distribution $\mathcal{U}$ over $\mathbb{A}$. Thus the n collaborators are viewed as n independent trials from $\mathcal{U}$, where each author $a_i \in \mathbb{A}$ has probability $1/M$ to be chosen[4]. Since one trial is reserved for a_1, only $n-1$ trials are really random. Suppose we have known e's ambiguity $\kappa(e)$. Then in each trial, choosing another author with name e has probability $\frac{\kappa(e)-1}{M-1} \approx \frac{\kappa(e)-1}{M}$.

The probability that no other collaborator of b has name e is:

$$\Pr(H_1^*|b) = (\frac{M-\kappa(e)}{M})^{n-1} \approx 1 - \frac{(n-1)\kappa(e)}{M}, \tag{6}$$

considering $\kappa(e) \ll M$. H_1^* means that for any pair of clusters C_1 and C_2, H_1 holds. So $H_1^* \implies H_1$, and $\Pr(H_1^*|b) \leq \Pr(H_1|b)$.

But we do not know n, the actual number of collaborators of b. We only know b has collaborated with $|\operatorname{co}(b)|$ **names**. So $n \geq |\operatorname{co}(b)|$. We can obtain n's expectation $\mathrm{E}[n]$ as n's estimation:

$$\mathrm{E}[n] \approx \frac{M(|\operatorname{co}(b)|-1)}{M - \sum_{e_i \in \operatorname{co}(b)} (\kappa(e_i)-1)}, \tag{7}$$

where $\kappa(e_i)$ is approximated by $\hat{\kappa}(e)$ in Section 6, and $M \approx \sum_{e \in \mathbb{A}} \hat{\kappa}(e)$.

Strong evidential coauthors require $\Pr(H_1|b) \geq \theta_{co}$. Combining this with Eq. (6), we obtain

$$n \leq \frac{(1-\theta_{co})M}{\kappa(e)} + 1. \tag{8}$$

The right-hand value of Eq. (8) is a threshold value to partition authors into two groups: one contains authors who have fewer coauthors than the threshold, and thus provide strong evidence; the other contains authors who have more coauthors than the threshold and thus offer weak evidence.

Given two clusters C_1 and C_2, if there is one shared strong evidential coauthors, then we see enough evidence supporting H_1, and then we merge them. Otherwise all shared coauthors are weak evidential. We use CSLR to see how likely the two coauthor sets are drawn from the same distribution. If $\Lambda(\operatorname{co}(C_1), \operatorname{co}(C_2)) > 1$, we merge C_1 and C_2.

[3] $\mathbb{A}$ includes all authors in the DBLP dump.

[4] The n trials is without replacement. The probability is approximated by trials with replacement. This approximation is good, since $n \ll M$.

5.3 Stage 2: Merging by Venue Set and Title Set

Consider a pair of clusters C_1 and C_2 with venue sets V_1, V_2, and title sets T_1, T_2. We denote the Venue Set Similarity by $\text{sim}_V(V_1, V_2)$, and Title Set Similarity by $\text{sim}_T(T_1, T_2)$. These two similarity measures are heterogeneous metrics, and we multiply them to compute the combined similarity:

$$\text{sim}(C_1, C_2) = \text{sim}_V(V_1, V_2) \cdot \text{sim}_T(T_1, T_2). \tag{9}$$

As the ambiguity $\kappa(e)$ of an author e increases, there are more and more authors working in the same subfields and publishing in the same venues. Therefore the clustering threshold in this stage, denoted by θ_c, should increase monotonically with $\kappa(e)$. We set θ_c as a linear function of $\hat{\kappa}(e)$:

$$\theta_c(\hat{\kappa}(e)) = 0.2 \cdot \max(1, \frac{1}{5}\hat{\kappa}(e)) \tag{10}$$

Due to space limitations, the technical details of using CSLR to compute the similarity $\text{sim}_V(V_1, V_2)$ and using BoW to compute $\text{sim}_T(T_1, T_2)$ are omitted here, and can be found in the full version of this paper ([6]).

Next we briefly introduce the idea of computing the two similarities.

Venue Set Expansion and Similarity. We use CSLR to compare two venue sets. But CSLR treats different outcomes as disparate and their correlations are not considered. Often two venue sets do not share common venues, but the venues are correlated, such as "TKDE" in one set, and "CIKM" in the other. They still favor (to certain degree) the hypothesis that the two clusters are from the same author. In this case, CSLR returns a very small likelihood ratio.

To remedy this problem, before computing CSLR, we expand each venue set with correlated venues first. Now a venue set {TKDE: 2, CIKM: 3} could become {TKDE: 2, CIKM: 3, ICDM: 1, KDD: 0.5}, and the CSLR value between it and another set {ICDM: 3, KDD: 1} will become reasonably large.

The idea is to predict the frequencies of absent but correlated venues of a set, based on observed venues, and then add the predicted $\{venue : frequency\}$ pairs into that set. The correlated venues are mined using *linear regression* on the 1.5 million DBLP papers.

We denote the expanded venue set of V_i as $\tilde{V}_i$, then $\text{sim}_V(V_1, V_2) = \Lambda(\tilde{V}_1, \tilde{V}_2)$.

Title Set Similarity Based on Unigram BoW. We adopt the traditional unigram BoW model to represent two title sets and calculate their similarity. The similarity $\text{sim}_T(T_1, T_2)$ is the weighted sum of shared unigrams[5]. The weighting scheme is a variant of TF*IDF, which regards all the titles of an author as a single document when calculating the Inverse Document Frequency (IDF).

[5] Words in the titles are so sparse and diverse that even if two title sets belong to the same author, the two corresponding sets of words are usually not drawn from the same distribution, and thus CSLR does not fit in here.

6 Name Ambiguity Estimation

We present a statistical method to estimate the ambiguity $\kappa(e)$ of each focus name e. The estimation $\hat{\kappa}(e)$ is used in (8) and (10). In addition, it plays two other roles: First, it is one of the stop criteria of the clustering. Once we reach $\hat{\kappa}(e)$ clusters, we should stop merging. Note the clustering may stop before the number of clusters becomes $\hat{\kappa}(e)$ due to other criteria. Second, if $\hat{\kappa}(e)$ is much less than 1, it means name e is rare, and it is highly possible that only one person has this name, regardless how many papers is authored by e. For example, in our dataset, 448 papers have author name *Jiawei Han*. We assert that all of them are by the same person, given that *Jiawei Han*'s estimated ambiguity is 0.29.

Our method is inspired by the "Ambiguity Estimate" intuition in [2]. Our estimation only needs the names statistics in a digital library.

In the digital library names in a given culture usually have a fixed number of parts. For example in DBLP, a Chinese name usually has 2 parts (e.g., "Xiaofeng" and "Wang" for name "Xiaofeng Wang"). Suppose that these parts were chosen roughly independently with each other. Thus we can estimate the probability of each option of each part, and then the probability of a full name is the joint probability of its parts.

We formulate the case of 3-part names as an example. Suppose a name e in a given culture consists of a given name $G(e)$, a middle name $M(e)$ and a family name $F(e)$, i.e., $e = G(e)+M(e)+F(e)$, where "+" means string concatenation.

For any name e in this culture, we assume $G(e)$, $M(e)$ and $F(e)$ are drawn independently from 3 categorical distributions Cat_G, Cat_M and Cat_E, respectively. Then $\Pr(e) = \Pr(G(e))\Pr(M(e))\Pr(F(e))$.

The parameters of Cat_G, Cat_M and Cat_E are estimated using MLE. Take Cat_G as an example. Let $\mathbb{E}$ be the set of all names in this culture, and $\mathbb{G}$ be the set of all given names in this culture,

$$\forall g \in \mathbb{G}, \quad \Pr(G(e) = g) \approx \frac{\sum_{e\in\mathbb{E}, G(e)=g} \kappa(e)}{\sum_{\forall e\in\mathbb{E}} \kappa(e)}. \tag{11}$$

Noticing $\sum_{\forall e\in E} \kappa(e)$ is the total number of different authors in this culture, the MLE of the instances (i.e., ambiguity) of name e in the DBLP author set is:

$$\hat{\kappa}(e) = \Pr(G(e))\Pr(M(e))\Pr(F(e)) \sum_{\forall e\in E} \kappa(e). \tag{12}$$

We do not know $\kappa(e)$, and thus we use $\hat{\kappa}(e)$ in place of $\kappa(e)$, and evaluate (11) and (12) iteratively, until $\hat{\kappa}(e)$ converges. It is possible that $\hat{\kappa}(e) < 1$ (a rare name), so during the iteration, we round $\hat{\kappa}(e)$ to 1 if $\hat{\kappa}(e) < 1$. Specifically,

1. Initially, $\forall e, \hat{\kappa}_0(e) = 1$;
2. In the $(i + 1)$-th iteration, we plug $\max(\hat{\kappa}_i(e), 1)$ for $\kappa(e)$ into (11) and (12), evaluate them and get $\hat{\kappa}_{i+1}(e)$. Repeat this step until $|\sum_{\forall e} \hat{\kappa}_{i+1}(e) - \sum_{\forall e} \hat{\kappa}_i(e)| \le \epsilon_m$, where ϵ_m is a small number to measure the convergence.

When the estimation converges at the n-th iteration, we round $\hat{\kappa}_n(e)$ up to 1 and get $\hat{\kappa}(e)$. If we want to check the rarity of a name, we use $\hat{\kappa}_n(e)$ directly.

Table 2. Statistics of Data Set 1*

Name e	**#Pubs**	$\kappa(e)$	$\hat{\kappa}(e)$
Hui Fang	9	3	1.62
Ajay Gupta	16	4	n/a
Joseph Hellerstein	151	2	n/a
Rakesh Kumar	36	2	n/a
Michael Wagner	29	5	n/a
Bing Liu	89	6	6.91
Jim Smith	19	3	n/a
Lei Wang	55	13 (31)	22.34
Wei Wang	140	14 (57)	49.43
Bin Yu	44	5 (11)	8.7

Table 3. Statistics of Data Set 2

Name e	**#Pubs**	$\kappa(e)$	$\hat{\kappa}(e)$
Hui Fang	45	8	6.8
Ajay Gupta	25	8	n/a
Joseph Hellerstein	234	2	n/a
Rakesh Kumar	104	8	n/a
Michael Wagner	61	16	n/a
Bing Liu	192	23	21.0
Jim Smith	54	5	n/a
Lei Wang	400	144	104.6
Wei Wang	833	216	254.2
Bin Yu	102	18	17.3

* [12] removed authors who have only one paper from their data set. So for the last three names in Table 2, [12] reported much smaller ambiguities than the real values, which are given in the parentheses.

Note the name-part independence assumption holds only among names in a given culture. Given names from one culture and family names from another culture are usually anti-correlated, for example "Jacob Li" is a very rare combination. So Ambiguity Estimation should be conducted culture-wise. For names in a culture which are too few in the digital library to form a large enough sample, external demographic data could be incorporated to get better estimation.

7 Experimental Results

7.1 Experimental Setting

Data Set. Two test sets are used. For fairness of comparison, both use the same set of names as in [12]. Papers written by these names in DBLP are extracted for disambiguation.

Set 1 is the same dataset as that used in [12]. Its statistics are listed in Table 2. This data set was extracted from a 2006 dump of DBLP.

Set 2 is extracted from a January 2011 dump of DBLP. Each name corresponds to many more papers (and bigger ambiguity, as more authors with these names publish) in Set 2 than Set 1. Their statistics are in Table 3. All these papers were hand-labeled and available at the URL given in Section 1.

As a part of our experiments, we test Ambiguity Estimation on Chinese author names, and list the results on names in the test set in Tables 2 and 3. Set 1 was built at the beginning of year 2006 ([12]), so we use the DBLP statistics before 2006 to estimate these ambiguities. Set 2 contains all authors and papers in DBLP till January 2011, and we use the whole DBLP statistics to estimate these ambiguities. The actual ambiguities $\kappa(e)$ are obtained by hand-labeling.

For Chinese names, our method gives a reasonable estimation: $\hat{\kappa}(e) \in (0.5\kappa(e), 1.5\kappa(e))$. We have not estimated the ambiguities of names in other cultures. But usually their ambiguities are small (below 30) and we set all of them to 2. Experiments show such inaccuracy does not impair the performance of our system noticeably.

Evaluation. As in [12,8], we use *Pairwise Precision*, *Pairwise Recall*, and *Pairwise F1* scores to evaluate the performance of our method and other methods. Specifically, any two papers that are annotated with the same label in the ground truth are called a correct pair, and any two papers that are predicted with the same label (if they are grouped in the same cluster, we also call they have the same label) by a system but are labeled differently in the ground truth are called a wrong pair. Note the counting is for pairs of papers with the same label (either predicted or labeled) only. Thereafter, we define the three scores:

$$\text{Prec} = \frac{\#\,\text{PairsCorrectlyPredicted}}{\#\,\text{TotalPairsPredicted}} \qquad \text{Rec} = \frac{\#\,\text{PairsCorrectlyPredicted}}{\#\,\text{TotalCorrectPairs}}$$

$$\text{F1} = \frac{2 \times \text{Prec} \times \text{Rec}}{\text{Prec} + \text{Rec}}$$

Experimental Details. We evaluated one baseline, denoted by Jac, which uses Jaccard Coefficient for coauthor/venue sets, the unigram BoW based similarity for title sets, and Eq. (10) as its clustering threshold. The optimal Jaccard Coefficient thresholds for coauthor sets and venue sets are different. We tested Jac with different thresholds, and chose the thresholds for coauthor sets and venue sets that produce the highest macro-average F1 scores, respectively. The best thresholds are 0.03 for coauthor sets, and 0.04 for venue sets.

We compared our method with two representative methods: DISTINCT ([12]) and Arnetminer ([11]). We acquired the original source code of DISTINCT. DISTINCT uses randomly generated training sets, and in different runs its performance varies greatly. Moreover, DISTINCT does not have a mechanism to determine a clustering threshold for a given name. Instead it tries 12 different thresholds between $[0, 0.02]$. For each name, different thresholds lead to disparate performance. So we ran DISTINCT 10 times and averaged its scores at each threshold, and then took the threshold that gives the highest macro-average F1 score over all names, as the chosen threshold (0.002 for Set 1, 0.001 for Set 2). Additionally, we crawled the disambiguation pages of these 10 names from `http://arnetminer.org/` on March 12, 2012, and extracted the disambiguation results. These results are generated by the up-to-date work of [11]. As Arnetminer contains papers newer than the release date of our DBLP dump, we discarded papers that are not in our data sets.

We refer to our own method as CSLR. It has 3 important parameters: θ_t, which controls the degree of toleration; θ_{co}, which controls the decision threshold between strong/weak-evidential coauthors; and $\theta_c(\hat{\kappa}(e))$, which controls when to stop the second-stage clustering. They are tuned on a development set of 5 names: *Tao Peng*, *Peng Cheng*, *David Jensen*, *Xiaodong Wang*, and *Gang Wu*.

7.2 Experimental Results and Discussion

The results for all methods are shown in Table 4 and 5. For each method, the most important measure, the macro-average F1 score over all names, is underlined. On both sets, CSLR significantly outperforms all the other methods.

Table 4. Comparison of Performance on Set 1

Name	Jac			Arnetminer			DISTINCT			Our (CSLR)		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Hui Fang	**100.0**	**100.0**	**100.0**	55.6	**100.0**	71.4	85.6	**100.0**	88.7	**100.0**	**100.0**	**100.0**
Ajay Gupta	**100.0**	93.1	96.4	**100.0**	**100.0**	**100.0**	67.7	94.5	78.8	**100.0**	93.1	96.4
Joseph Hellerstein	50.7	83.9	63.2	97.4	**97.4**	**97.4**	92.4	80.6	84.6	**100.0**	69.7	82.1
Rakesh Kumar	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Michael Wagner	**100.0**	64.0	78.1	**100.0**	33.7	50.5	90.1	**96.2**	**92.9**	**100.0**	64.0	78.1
Bing Liu	**99.8**	84.5	**91.5**	86.2	79.8	82.9	86.5	82.0	83.6	91.8	**87.0**	89.4
Jim Smith	**100.0**	83.1	90.8	**100.0**	84.5	91.6	95.6	**91.7**	**93.3**	**100.0**	87.3	93.2
Lei Wang	**100.0**	71.2	**83.2**	59.4	**94.2**	72.9	42.5	75.0	51.8	**100.0**	63.3	77.6
Wei Wang	**60.5**	83.7	**70.2**	28.1	98.5	43.8	31.0	**98.8**	47.1	59.3	72.4	65.2
Bin Yu	70.7	64.7	67.6	87.8	**95.3**	**91.4**	77.1	89.2	81.3	**98.8**	68.5	80.9
Avg. (macro-F1)	88.2	82.8	84.1	81.5	88.4	80.2	76.9	**90.8**	80.2	**95.0**	80.5	**86.3**

Table 5. Comparison of Performance on Set 2

Name	Jac			Arnetminer			DISTINCT			Our (CSLR)		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Hui Fang	**100.0**	68.8	81.5	59.1	63.7	61.3	81.3	**97.9**	88.0	**100.0**	78.9	**88.2**
Ajay Gupta	**96.0**	47.0	63.1	60.0	65.4	62.6	65.3	**87.9**	**74.2**	**96.0**	39.6	56.1
Joseph Hellerstein	52.8	80.5	63.7	94.5	**95.9**	**95.2**	92.3	89.5	90.0	**100.0**	79.6	88.6
Rakesh Kumar	**100.0**	89.0	94.2	98.4	89.3	93.7	89.9	96.0	92.5	99.9	**97.8**	**98.8**
Michael Wagner	**92.8**	59.4	72.4	55.6	36.7	44.2	67.4	**98.2**	**79.1**	88.1	64.6	74.6
Bing Liu	97.8	67.0	79.5	75.7	67.2	71.2	83.0	**84.7**	83.3	**98.1**	74.7	**84.8**
Jim Smith	**100.0**	44.1	61.2	88.6	45.1	59.7	94.8	**87.8**	**90.0**	**100.0**	48.8	65.6
Lei Wang	30.0	79.8	43.6	18.1	83.1	29.8	29.3	85.9	42.4	**78.1**	**87.6**	**82.6**
Wei Wang	40.2	77.0	52.8	9.7	**88.2**	17.5	25.8	84.2	38.9	**81.0**	71.8	**76.1**
Bin Yu	70.6	42.8	53.3	72.4	**62.2**	**66.9**	54.0	62.0	57.0	**88.0**	49.1	63.0
Avg. (macro-F1)	78.0	65.5	66.5	63.2	69.7	60.2	68.3	**87.4**	73.5	**92.9**	69.2	**77.8**

On Set 1 DISTINCT has a lower macro-average F1 score than that reported in [12]. We think it is partly due to the random nature of DISTINCT when it chooses a random training set to trains the feature weights. But since we have run DISTINCT for consecutive 10 times, we think the average scores truly reflect its performance in practice without ground truth to select the best trained weights.

On Set 2 Arnetminer has a sudden performance drop compared to its performance on Set 1. One important "culprit" is its precision on *Wei Wang* is extremely low. As we can see in the actual disambiguation result online at `http://arnetminer.org/`, 727 papers are credited to the professor at UNC, among which we believe only < 200 papers are authored by her. The reason might be Arnetminer merges clusters based on a few weak evidential coauthors.

The baseline Jac performs well on Set 1. This may ascribe to two factors: 1) It uses the optimal Jaccard Coefficient thresholds, which are impossible to obtain in practice without ground truth; 2) It uses the same estimated name ambiguity to set the clustering threshold. However, Jac's performance plunges on Set 2 where the ambiguity of each name is larger. This contrast suggests the adverse effect of the inaccuracy of Jaccard Coefficient intensifies as the ambiguity grows.

Compared to other methods, our system has slightly lower recall, but much higher precision. We think a major reason is that CSLR returns a high similarity only when two clusters follow similar distributuions. Sometimes clusters of papers by the same author are drastically different (e.g., very few shared venues and

shared terms in titles), and it is difficult even for a human to decide whether they belong to the same author. From a user's perspective, it is often more frustrating to see papers of different authors are mixed up (low precision), than to see papers of the same author are split into smaller clusters (low recall).

8 Conclusions and Future Work

In this paper, we present a novel categorical set similarity measure named CSLR for two sets which both follow categorical distributions. It is applied in Author Name Disambiguation to measure the similarity between two venue sets or coauthor sets. It is verified to be better than the widely used *Jaccard Coefficient*. We have also proposed a novel method to estimate the distinct author number of each name, which gives reasonable estimation. Our experiments show that our system clearly outperforms other methods of comparison.

We envision broad applications of CSLR since it is a general categorical set similarity measure. In scenarios such as Social Networks and Natural Language Processing, an entity often has a set of contextual features. Often these features have categorical values, and two entities are similar iff these sets follow similar categorical distributions. Some previous work used Jaccard Coefficient etc. as the similarity measures. We expect CSLR will perform better than them.

References

1. Agresti, A.: Categorical data analysis. Wiley series in probability and statistics. Wiley-Interscience (2002)
2. Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. ACM Trans. Knowl. Discov. Data 1 (March 2007)
3. Cota, R.G., Ferreira, A.A., Nascimento, C., Gonalves, M.A., Laender, A.H.F.: An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations. J. Am. Soc. Inf. Sci. Technol. 61(9), 1853–1870 (2010)
4. Gretton, A., Borgwardt, K., Rasch, M., Schlkopf, B., Smola, A.: A kernel method for the two sample problem. In: NIPS, vol. 19, pp. 513–520. MIT Press (2007)
5. Han, H., Giles, L., Zha, H., Li, C., Tsioutsiouliklis, K.: Two supervised learning approaches for name disambiguation in author citations. In: JCDL 2004. ACM (2004)
6. Li, S., Cong, G., Miao, C.: Supplementary material to author name disambiguation using a categorical distribution similarity, http://git.io/namedis
7. Pereira, D.A., Ribeiro-Neto, B., Ziviani, N., Laender, A.H., Gonçalves, M.A., Ferreira, A.A.: Using web information for author name disambiguation. In: JCDL 2009. ACM (2009)
8. Tang, J., Fong, A.C., Wang, B., Zhang, J.: A unified probabilistic framework for name disambiguation in digital library. IEEE TKDE 99 (2011) (preprints)
9. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: KDD 2008. ACM (2008)
10. Torvik, V.I., Smalheiser, N.R.: Author name disambiguation in medline. ACM Trans. Knowl. Discov. Data 3, 11:1–11:29 (2009)
11. Wang, X., Tang, J., Cheng, H., Yu, P.S.: Adana: Active name disambiguation. In: ICDM 2011 (2011)
12. Yin, X., Han, J., Yu, P.S.: Object distinction: Distinguishing objects with identical names by link analysis. In: ICDE 2007 (2007)

Lifted Online Training of Relational Models with Stochastic Gradient Methods

Babak Ahmadi[1], Kristian Kersting[1,2,3], and Sriraam Natarajan[3]

[1] Fraunhofer IAIS, Knowledge Discovery Department, Sankt Augustin, Germany
[2] University of Bonn, Institute of Geodesy and Geoinformation, Bonn, Germany
[3] Wake Forest University, School of Medicine, Winston-Salem, USA

Abstract. Lifted inference approaches have rendered large, previously intractable probabilistic inference problems quickly solvable by employing symmetries to handle whole sets of indistinguishable random variables. Still, in many if not most situations training relational models will not benefit from lifting: symmetries within models easily break since variables become correlated by virtue of depending asymmetrically on evidence. An appealing idea for such situations is to train and recombine local models. This breaks long-range dependencies and allows to exploit lifting within and across the local training tasks. Moreover, it naturally paves the way for online training for relational models. Specifically, we develop the first lifted stochastic gradient optimization method with gain vector adaptation, which processes each lifted piece one after the other. On several datasets, the resulting optimizer converges to the same quality solution over an order of magnitude faster, simply because unlike batch training it starts optimizing long before having seen the entire mega-example even once.

1 Introduction

Statistical relational models, see [1, 2] for overviews, have recently gained popularity in the machine learning and AI communities since they provide powerful formalisms to compactly represent complex real-world domains. Unfortunately, computing the exact gradient in such models and hence learning the parameters with exact maximum-likelihood training using current optimization methods like conjugate gradient and limited-memory BFGS is often not feasible as it requires computing marginal distributions of the entire underyling graphical model. Since inference is posing major computational challenges one has to resort to approximate learning.

One attractive avenue to scale relational learning is based on lifted message-passing approaches [3, 4]. They have rendered large, previously intractable probabilistic inference problems quickly (often approximately) solvable by employing symmetries to handle whole sets of indistinguishable random variables. Still, in most situations training relational models will not benefit from lifting:

> **(Limitation 1)** *Symmetries within a model easily break since variables become correlated by virtue of depending asymmetrically on evidence.*

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 585–600, 2012.

Because of this, lifting produces new models that are often not far from propositionalized, therefore canceling the benefits of lifting for training. Moreover, in relational learning we often face a single mega-example [5] only, a single large set of inter-connected facts. Consequently, many if not all standard statistical learning methods do not naturally carry over to the relational case. Consider e.g. stochastic gradient methods. Similar to the perceptron method [6], stochastic gradient descent algorithms update the weight vector in an online setting. We essentially assume that the training examples are given one at a time. The algorithms examine the current training example and then update the parameter vector accordingly. They often scale sub-linearly with the amount of training data, making them very attractive for large training data as targeted by statistial relational learning. Empirically, they are even often found to be more resilient to errors made when approximating the gradient. Unfortunately, stochastic gradient methods do not naturally carry over to the relational cases:

> **(Limitation 2)** *Stochastic gradients coincide with batch gradients in the relational case since there is only a single mega-example.*

In this paper, we demonstrate how to overcome both limitations.

To do so, we shatter the full model into pieces. In each iteration, we train the pieces independently and re-combine the learned parameters from each piece. This overcomes **limitation 1** by breaking long-range dependencies and allows one — as we will show — to exploit lifting across the local training tasks. It also paves the way for online training — as we will show — of relational models since we can treat (mini-batches of) pieces as training examples and process one piece after the other, hence overcoming **limitation 2**. Based on this insight, we develop our main algorithmic contribution: the first lifted online training approach for relational models using a stochastic gradient optimization method with gain vector adaptation based on natural gradients. As our experimental evaluation demonstrates, it already results in considerable efficiency gains, simply because unlike batch training it starts optimizing long before having seen the entire mega-example even once. However, we can do considerably better. The way we shatter the full model into pieces greatly effects the learning quality. Important influences between variables might get broken. To overcome this, we randomly grow relational piece patterns that form trees. Our experimental results show that *tree pieces* can balance well lifting and quality of the online training.

We proceed as follows. After touching upon related work, we recap Markov logic networks, the probabilistic relational framework we focus on for illustration purpose. Then, we develop the stochastic relational gradient framework. Before concluding, we present our experimental evaluation.

2 Related Work

Our work aims at combining stochastic gradient methods for online training, relational learning, and lifted inference hence is related to several lines of research.

Local training is well known for propositional graphical models. Besag [7] presented a pseudolikelihood (PL) approach for training an Ising model with a

rectangular array of variables. PL, however, tends to introduce a bias and is not necessarily a good approximation of the true likelihood with a smaller number of samples. In the limit, however, the maximum pseudolikelihood coincides with that of the true likelihood [8]. Hence, it is a very popular method for training models such as Conditional Random Fields (CRF) where the normalization can become intractable while PL requires normalizing over only one node. An alternative approach is to decompose the factor graph into tractable subgraphs (or pieces) that are trained independently [9], as also follows in the present paper. This *piecewise training* can be understood as approximating the exact likelihood using a propagation algorithm such as BP. Sutton and McCallum [9] also combined the two ideas of PL and piecewise training to propose piecewise pseudolikelihood (PWPL) which inspite of being a double approximation has the benefit of being accurate like piecewise and scales well due to the use of PL. Another intuitive approach is to compute approximate marginal distributions using a global propagation algorithm like BP, and simply substitute the resulting beliefs into the exact ML gradient [10], which will result in approximate partial derivatives. Similarly, the beliefs can also be used by a sampling method such as MCMC where the true marginals are approximated by running an MCMC algorithm for a few iterations. Such an approach is called constructive divergence [11] and is a popular method for training CRFs.

All the above methods were originally developed for propositional data while real-world data is inherently noisy and relational. Statistical Relational Learning (SRL) [1, 2] deals with uncertainty and relations among objects. The advantage of relational models is that they can succinctly represent probabilistic dependencies among the attributes of different related objects leading to a compact representation of learned models. While relational models are very expressive, learning them is a computationally intensive task. Recently, there have been some advances in learning SRL models, especially in the case of Markov Logic Networks [12–14]. Algorithms based on functional-gradient boosting [15] have been developed for learning SRL models such as Relational Dependency Networks [16], and Markov Logic Networks [14]. Piecewise learning has also been pursued already in SRL. For instance, the work by Richardson and Domingos [17] used pseudolikelihood to approximate the joint distribution of MLNs which is inspired from the local training methods mentioned above. Though all these methods exhibit good empirical performance, they apply the closed-world assumption, i.e., whatever is unobserved in the world is considered to be false. They cannot easily deal with missing information. To do so, algorithms based on classical EM [18] have been developed for ProbLog, CP-logic, PRISM, probabilistic relational models, Bayesian logic programs [19–23], among others, as well as gradient-based approaches for relational models with complex combining rules [24, 25]. All these approaches, however, assume a *batch* learning setting; they do not update the parameters until the entire data has been scanned. In the presence of large amounts of data such as relational data, the above method can be wasteful. Stochastic gradient methods as considered in the present paper, on the other hand, are online and scale sub-linearly with the amount of

training data, making them very attractive for large data sets. Only Huynh and Mooney [26] have recently studied online training of MLNs. Here, training was posed as an online max margin optimization problem and a gradient for the dual was derived and solved using incremental-dual-ascent algorithms. They, however, do not employ lifted inference for training and also make the closed-world assumption.

3 Markov Logic Networks

We develop our lifted online training method within the framework of Markov logic networks [17] but would like to note that it naturally carries over to other relational frameworks. A Markov logic network (MLN) is defined by a set of first-order formulas (or clauses) F_i with associated weights w_i, $i \in \{1, \ldots, k\}$. Together with a set of constants $C = \{C_1, C_2, \ldots, C_n\}$ it can be grounded, i.e. the free variables in the predicates of the formulas F_i are bound to be constants in C, to define a Markov network. This ground Markov network contains a binary node for each possible grounding of each predicate, and a feature for each grounding f_k of each formula. The joint probability distribution of an MLN is given by $P(X = x) = Z^{-1} \exp\left(\sum_i^{|F|} \theta_i n_i(x)\right)$ where for a given possible world x, i.e. an assignment of all variables X, $n_i(x)$ is the number of times the ith formula is evaluated *true* and Z is a normalization constant.

The standard parameter learning task for Markov Logic networks can be formulated as follows. Given a set of training instances $D = \{D_1, D_2, \ldots D_M\}$ each constisting of an assignment to the variables in X the goal is to output a parameter vector θ specifying a weight for each $F_i \in F$. Typically, however, a single mega-example [5] is only given, a single large set of inter-connected facts. For the sake of simplicity we will sometimes denote the mega-example simply as E. To train the model, we can seek to maximize the log-likelihood function $\log P(D \mid \theta)$ given by $\ell(\theta, D) = \frac{1}{n} \sum_D \log P_\theta(X = x_{D_n})$. The likelihood, however, is computationally hard to obtain. A widely-used alternative is to maximize the pesudo-log-likelihood instead i.e., $\log P^*(X = x \mid \theta) = \sum_{l=1}^{n} \log P_\theta(X = x_l | MB_x(X_l))$ where $MB_x(X_l)$ is the state of the Markov blanket of X_l in the data, i.e. the assignment of all variables neighboring X_l. In this paper, we resort to likelihood maximization. No matter which objective function is used, one typically runs a gradient-descent to train the model. That is, we start with some initial parameters θ_0 — typically initialized to be zero or at random around zero — and update the parameter vector using $\theta_{t+1} = \theta_t - \eta_t \cdot g_t$. Here g_t denotes the gradient of the likelihood function and is given by:

$$\partial \ell(\theta, D)/\partial \theta_k = n_k(D) - M\mathbf{E}_{\mathbf{x} \sim P_\theta}[n_k(\mathbf{x})] \tag{1}$$

This gradient expression has a particularly intuitive form: the gradient attempts to make the feature counts in the empirical data equal to their expected counts relative to the learned model. Note that, to compute the expected feature counts, we must perform inference relative to the current model. This inference step

must be performed at every step of the gradient process. In the case of partially observed data we cannot simply read-off the feature counts in the empirical data and have to perform inference there as well. Consequently, there is a close interaction between the training approach and the inference method employed for training.

4 Lifted Online Training

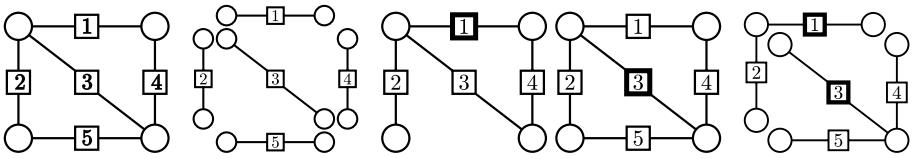
Lifted Belief propagation (LBP) approaches [4, 27] have recently drawn a lot of attention as they render large previously intractable models quickly solvable by exploiting symmetries. Such symmetries are commonly found in first-order and relational probabilistic models that combine aspects of first-order logic and probability. Instantiating all ground atoms from the formulae in such models induces a standard graphical model with symmetric, repeated potential structures for all grounding combinations. To exploit the symmetries, LBP approaches automatically group nodes and potentials of the graphical model into supernodes and superpotentials if they have identical computation trees (i.e., the tree-structured unrolling of the graphical model computations rooted at the nodes). LBP then runs a modified BP on this lifted (clustered) network simulating BP on the propositional network obtaining the same results. When learning parameters of a given model for a given set of observations, however, the presence of evidence on the variables mostly destroys the symmetries. This makes lifted approaches virtually of no use if the evidence is non symmetrical.

In the fully observed case, this may not be a major obstacle since we can simply count how often a clause is true. Unfortunately, in many real-world domains, the mega-example available is incomplete, i.e., the truth values of some ground atoms may not be observed. For instance in medical domains, a patient rarely gets all of the possible tests. In the presence of missing data, however, the maximum likelihood estimate typically cannot be written in closed form. It is a numerical optimization problem, and typically involves nonlinear, iterative optimization and multiple calls to a relational inference engine as subroutine.

Since efficient lifted inference is troublesome in the presence of partial evidence and most lifted approaches basically fall back to the ground variants we need to seek a way to make the learning task tractable. An appealing idea for efficiently training large models is to divide the model into pieces that are trained independently and to exploit symmetries across multiple pieces for lifting.

4.1 Piecewise Shattering

In piecewise training, we decompose the mega-example and its corresponding factor graph into tractable but not necessarily disjoint subgraphs (or pieces) $\mathcal{P} = \{p_1, \ldots, p_k\}$ that are trained independently [28]. Intuitively, the pieces turn the single mega-example into a set of many training examples and hence pave the way for online training. This is a reasonable idea since in many applications, the local information in each factor alone is already enough to do well at predicting



(a) Orig. model (b) Depth $d = 0$ (c) Depth $d = 1$ for f_1 and f_3 (d) Trees $d = 1$

Fig. 1. Schematic factor-graph depiction of the difference between likelihood (a), standard piecewise (b,c) and treewise training (d). Likelihood training considers the whole mega-example, i.e., it performes inference on the complete factor graph induced over the mega-example. Here, circles denote random variables, and boxes denote factors. Piecewise training normalizes over one factor at a time (b) or higher-order, complete neighbourhoods of a factor (c) taking longer dependcies into account, here shown factors f_1 and f_3. Treewise training (d) explores the spectrum between (b) and (c) in that it also takes longer dependecies into account but does not consider complete higher neighbourhoods; shown for tree features for factors f_1 and f_3. In doing so it balances complexity and accuracy of inference.

the outputs. The parameters learned locally are then used to perform global inference on the whole model.

More formally, at training time, each piece from $\mathcal{P} = \{p_1, \ldots, p_k\}$ has a local likelihood as if it were a separate graph, i.e., training example and the global likelihood is estimated by the sum of its pieces: $\ell(\theta, D) = \sum_{p_i \in \mathcal{P}} \ell(\theta|_{p_i}, D|_{p_i})$. Here $\theta|_{p_i}$ denotes the parameter vector containing only the parameters appearing in piece p_i and $D|_{p_i}$ the evidence for variables appearing in the current piece p_i. The standard piecewise decomposition breaks the model into a separate piece for each factor. Intuitively, however, this discards dependencies of the model parameters when we decompose the mega-example into pieces. Although the piecewise model helps to significantly reduce the cost of training the way we shatter the full model into pieces greatly effects the learning and lifting quality. Strong influences between variables might get broken. Consequently, we next propose a shattering approach that aims at keeping strong influence but still features lifting.

4.2 Relational Tree Shattering

Assume that the mega-example has been turned into a single factor graph for performing inference, cf. Fig. 1**(a)**. A factor graph is a bipartite graph and contains nodes representing random variables (denoted by circles) and factors (squares). It explicitly represents the factorization of the graphical model and there is an edge between a factor f_k and a node i iff variable X_i appears in f_k. Now, starting from each factor, we extract networks of depth d rooted in this factor. A local network of depth $d = 0$ thus corresponds to the standard piecewise model as shown in Fig. 1**(b)**, i.e. each factor is isolated in a separate piece. Networks of depth $d = 1$ contain the factor in which it is rooted and

Algorithm 1. RelTreeFinding: Relational Treefinding

Input: Set of clauses **F**, a mega example **E**, depth **d**, and discount $t \in [0, 1]$
Output: Set of tree pieces **T**

```
// Tree-Pattern Finding
```
1 Initialize the dictionary of tree patterns to be empty, i.e., $P = \emptyset$;
2 **for each** *clause* $F_i \in F$ **do**
3 Select a random ground instance f_j of F_i in E;
4 Initialize tree pattern for F_i, i.e., $P_i = \{f_j\}$;
 `// perform random walk in a breath-first manner starting in` f_j
5 **for** f_k = `BFS.`*next()* **do**
6 **if** *current_depth* $> d$ **then** break;
7 sample p uniformly from $[0, 1]$;
8 **if** $p > t^{|P_i|}$ **or** f_k *would induce a cycle* **then**
9 skip branch rooted in f_k in BFS ;
10 **else**
11 add f_k to P_i;
12 Variablilize P_i and add it to dictonary P;

```
// Construct tree-based pieces using the relational tree patterns
```
13 **for each** $f_j \in$ **E do**
14 Find $P_k \in P$ matching f_j, i.e., the tree pattern rooted in the clause F_k corresponding to factor f_j;
15 Unify P_k with f_j to obtain piece T_j and add T_j to T ;
16 **return** T;

all of its direct neighbors, Fig. 1**(c)**. Thus when we perform inference in such local models using say belief propagation (BP) the messages in the root factor of such a network resemble the BP messages in the global model up to the d-th iteration. Longer range dependencies are neglected. A small value for d keeps the pieces small and makes inference and hence training more efficient, while a large d is more accurate. However, it has a major weakness since pieces of densely connected networks may contain considerably large subnetworks, rendering the standard piecewise learning procedure useless.

To overcome this, we now present a shattering approach that randomly grows piece patterns forming trees. Formally, a tree is defined as a set of factors such that for any two factors f_1 and f_n in the set, there exists one and only one ordering of (a subset of) factors in the set f_1, f_2 , $\ldots f_n$ such that f_i and f_{i+1} share at least one variable, i.e. there are no loops. A tree of factors can then be generalized into a tree pattern, i.e., conjunctions of relational "clauses" by variablizing their arguments. For every clause of the MLN we thus form a tree by performing a random walk rooted in one ground instance of that clause. This process can be viewed as a form of relational pathfinding [29].

The relational treefinding is summarized in Alg. 1. For a given set of Clauses F and a mega example E the algorithm starts off by constructing a tree pattern for each clause F_i **(lines 1-12)**. Therefore, it first selects a random ground instance f_j **(line 3)** from where it grows the tree. Then it performs a

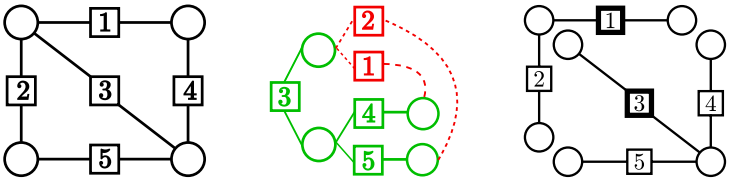


Fig. 2. Illustration of tree shattering: from the original model (left) we compute a tree piece (right). Starting from factor f_3, we randomly follow the tree-structured "unrolling"' of the graphical model rooted at f_3. Green shows that the factor has been included in the random walk while all red factors have been discarded. This results in the tree pattern for f_3 shown on the right hand side. A similar random walk generated the other shown tree pattern for f_1.

breadth-first traversal of the factors neighborhood and samples uniformly whether they are added to the tree or not **(line 7)**. If the sample p is larger than $t^{|P_i|}$, where $t \in [0, 1]$ is a discount threshold and $|P_i|$ the size of the current tree, or the factor would induce a cycle, the factor and its whole branch are discarded and skipped in the breadth-first traversal, otherwise it is added to the current tree **(lines 8-11)**. A small t basically keeps the size of the tree small while larger values for t allow for more factors being included in the tree. The procedure is carried out to a depth of at most d, and then stops growing the tree. This is then generalized into a piece-pattern by variablizing its arguments **(line 12)**. All pieces are now constructed based on these piece patterns. For f_j we apply the pattern P_k of clause F_k which generated the factor **(lines 13-15)**.

These *tree-based pieces* can balance efficiency and quality of the parameter estimation well. Reconsider the example from Fig. 1. Fig. 2 shows the tree rooted in the factor f_3 where green colors show that the factors have been included in the piece while all red factors have been discarded. The neighborhood of factor f_3 is traversed in a breadth-first manner, i.e., first its direct neighbors in random order. Assume we have reached factor f_4 first. We uniformly sample a $p \in [0, 1]$. It was small enough, e.g. $p = 0.3 < 0.9^1$ so f_4 is added to the tree. For f_2 we sample $p = 0.85 > 0.9^2$ so f_2 and its branch are discarded. For f_1 we sample $p = 0.5 < 0.9^2$ so f_1 could be added. If we added f_1, however, it would together with f_3 and f_4 form a cycle, so its branch is discarded. For f_5 we sample $p = 0.4 < 0.9^2$ so it is added to the tree. Note that now we cannot add any more edges without including cycles. In this way we can include longer range dependencies in our pieces without sacrificing efficiency. The connectivity of a piece and thereby its size can be controlled via the discount t. By forming tree patterns and applying them to all factors we ensure that we have a potentially high amount of lifting: *Since we have decomposed the model into smaller pieces, the influence of the evidence is limited to a shorter range and hence features lifting the local models.*

Moreover, we get an upper bound on the log partition function $A(\Theta)$. To see, this, we first write the original parameter vector Θ as a mixture of parameter vectors $\Theta(T_t)$ induced by the tractable subgraphs. For each edge in our

mega-example E, we add a non-spanning tree T_t which contains all the original vertices but only the edges present in t. With each tree T_t we associate an exponential parameter vector $\Theta(T_t)$. Let μ be a strictly positive probability distribution over the tractable subgraphs, such that the original parameter vector Θ can be written as a combination of per-tree-clause parameter vectors

$$\Theta = \sum_F \sum_t \mu_{t,F} \Theta(T_t) \ ,$$

where we have expressed parameter sharing among the ground instance of the clauses. Now using Jensen's inequality, we can state the following upper bound to the log partition function:

$$A(\Theta) = A(\sum_F \sum_t \mu_{t,F} \Theta(T_t)) = A(\sum_t \mu_t \Theta(T_t)) \leq \sum_t \mu_t A(\Theta(T_t)) \qquad (2)$$

with $\mu_t = \sum_F \mu_{t,F}$. Since the $\mu_{t,F}$ are convex, the μ_t are convex, too, and applying Jensen's inequality is safe. So we can follow Sutton and McCallum's [9] arguments. Namely, for tractable subgraphs and a tractable number of models the right-hand side of (2) can be computed efficiently. Otherwise it forms an optimization problem, which according to [30] can be interpreted as free energy and depends on a set of marginals and edge appearance probabilities, in our case the probability that an edge appears in a tree, i.e. is visited in the random walk. Also, it is easy to show that pieces of depth 0 are an upper bound to this bound since, we can apply Jensen's inequality again when breaking the trees into independent pathes from the root to the leaves.

Now, we show how to turn this upper bound into a lifted online training for relational models.

4.3 Lifted Stochastic Meta-descent

Stochastic gradient descent algorithms update the weight vector in an online setting. We essentially assume that the pieces are given one at a time. The algorithms examine the current piece and then update the parameter vector accordingly. They often scale sub-linearly with the amount of training data, making them very attractive for large training data as targeted by statistial relational learning. To reduce variance, we may form *mini-batches* consisting of several pieces on which we learn the parameters locally. In contrast to the propositional case, however, mini-batches have another important advantage: we can now make use of the symmetries within and *across* pieces for lifting.

More formally, the gradient in (1) is approximated by

$$\sum\nolimits_i \frac{1}{\#_i} \frac{\partial \ell(\theta, D_i)}{\partial \theta_k}, \qquad (3)$$

where the mega-example D is partitioned into pieces respectively mini-batches of pieces D_i. Here $\#_i$ denotes a per-clause normalization that counts how often each

clause appears in mini-batch D_i. This is a major difference to the propositional case and avoids "double counting" parameters. For example, let g_i be a gradient over the the mini-batch D_i. For a single piece we count how often a ground instance of each clause appears in the piece D_i. If D_i consists of more than one piece we add the count vector of all pieces together. For example, if for a model with 4 clauses the single piece mini-batch D_i has counts $(1, 3, 0, 2)$ the gradient is normalized by the respective counts. If the mini-batch, however, has an additional piece with counts $(0, 2, 1, 0)$ we normalize by the sum, i.e. $(1, 5, 1, 2)$.

Since the gradient involves inference per batch only, inference is again feasible and more importantly liftable as we will show in the experimental section. Consequently, we can scale to problem instances traditional relational methods can not easily handle. However, the asymptotic convergence of first-order stochastic gradients to the optimum can often be painfully slow if e.g. the step-size is too small. One is tempted to just employ standard advanced gradient techniques such as L-BFGS. Unfortunately most advanced gradient methods do not tolerate the sampling noise inherent in stochastic approximation: it collapses conjugate search directions [31] and confuses the line searches that both conjugate gradient and quasi-Newton methods depend upon. Gain adaptation methods like Stochastic Meta-Descent (SMD) overcome these limitations by using second-order information to adapt a per-parameter step size [32]. However, while SMD is very efficient in Euclidian spaces, Amari [33] showed that the parameter space is actually a Riemannian space of the metric C, the covariance of the gradients. Consequently, the ordinary gradient does not give the steepest direction of the target function. The steepest direction is instead given by the natural gradient, that is by $C^{-1}g$. Intuitively, the natural gradient is more conservative and does not allow large variances. If the gradients highly disagree in one direction, one should not take the step. Thus, whenever we have computed a new gradient g_t we integrate its information and update the covariance at time step t by the following expression:

$$C_t = \gamma C_{t-1} + g_t g_t^T \tag{4}$$

where $C_0 = 0$, and γ is a parameter that controls how much older gradients are discounted. Now, let each parameter θ_k have its own step size η_k. We update the parameter b

$$\theta_{t+1} = \theta_t - \boldsymbol{\eta}_t \cdot g_t \tag{5}$$

The gain vector $\boldsymbol{\eta}_t$ serves as a diagonal conditioner and is simultaneously adapted via a multiplicative update with the meta-gain μ:

$$\eta_{t+1} = \eta_t \cdot \exp(-\mu g_{t+1} \cdot v_{t+1}) \approx \eta_t \cdot \max(\frac{1}{2}, 1 - \mu g_{t+1} \cdot v_{t+1}) \tag{6}$$

where $v \in \Theta$ characterizes the long-term dependence of the system parameters on gain history over a time scale governed by the decay factor $0 \leq \lambda \leq 1$ and is iteratively updated by

$$v_{t+1} = \lambda v_t - \eta \cdot (g_t + \lambda C^{-1} v_t) \,. \tag{7}$$

Algorithm 2. Lifted Online Training of Relational Models

Input: Markov Logic Network $\mathbf{M}$, mega-example E, decay factors t, γ, and λ
Output: Parameter vector θ

```
// Generate mini-batches
```
1 Generate set of tree pieces $\mathcal{P}$ using RelTreeFinding;
2 Randomly form mini-batches $\mathcal{B} = \{B_1, \ldots, B_m\}$ each consisting of l pieces;

```
// Peform lifted stochastic meta-descent
```
3 Initialize θ and v_0 with zeros and the covariance matrix $\mathbf{C}$ to the zero matrix;
4 **while** *not converged* **do**
5 Shuffle mini-batches $\mathcal{B}$ randomly;
6 **for** $i = 1, 2, \ldots, m$ **do**
7 Compute gradient g for B_i using lifted belief propagation;
8 Update covariance matrix $\mathbf{C}$ using (4) or some low-rank variant;
9 Update parameter vector θ using (5) and the involved equations;
10 **return** θ;

To ensure a low computational complexity and a good stability of the computations, one can maintain a low rank approximation of C, see [34] for more details. Using per-parameter step-sizes considerably accelerates the convergence of stochastic natural gradient descent.

Putting everything together, we arrive at the lifted online learning for relational models as summarized in Alg. 2. That is, we form mini-batches of tree pieces **(lines 1-2)**. After initialization **(lines 3-4)**, we then perform lifted stochastic meta-descent **(lines 5-9)**. That is, we randomly select a mini-batch, compute its gradient using lifted inference, and update the parameter vector. Note that pieces and mini-batches can also be computed on the fly and thus its construction be interweaved with the parameter update. We iterate these steps until convergence, e.g. by considering the change of the parameter vector in the last l steps. If the change is small enough, we consider it as evidence of convergence. To simplify things, we may also simply fix the number of times we cycle through all mini-batches. This also allows to compare different methods.

5 Experimental Evaluation

Our intention here is to investigate the following questions: **(Q1)** Can we efficiently train relational models using stochastic gradients? **(Q2)** Are there symmetries within mini-batches that result in lifting? **(Q3)** Can relational treefinding produce pieces that balance accuracy and lifting well? **(Q4)** Is it even possible to achieve one-pass relational training?

To this aim, we implemented lifted online learning for relational models in Python. As a batch learning reference, we used *scaled conjugate gradient (SCG)* [35]. SCG chooses the search direction and the step size by using information from the second order approximation. Inference that is needed as a subroutine for the learning methods was carried out by lifted belief propagation

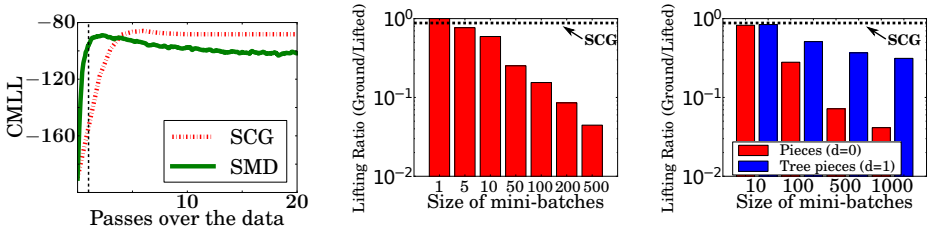


Fig. 3. "Passes over mega-example" vs. Test-CMLL for the Friends-and-Smokers **(left)** (the higher the better). lifted online learning has already learned before seeing the mega example even once (black vertical line). **(center)** Benefit of local training for lifting. Lifting ratio for varying mini-batch size versus the full batch model on the Friends-and-Smokers MLN. Clearly for a batch size of 1 there is no lifting but with larger mini-batch sizes there is more potential to lift the pieces within each batch; the size can be an order of magnitude smaller. **(right)** Lifting ratio for standard pieces vs. tree pieces on the Voting MLN. Due to rejoining of pieces, additional symmetries are broken and the lifting potential is smaller. However, the sizes of the models per mini-batch still gradually decrease with larger mini-batch sizes. (Best viewed in color)

(LBP) [4, 36]. For evaluation, we computed the *conditional marginal log-likelihood* (CMLL) [10], which is defined with respect to marginal probabilities. More precisely, we first divide the variables into two groups: X_{hidden} and $X_{observed}$. Then, we compute $\text{CMLL}(E) = \sum_{X \in X_{hidden}} \log P(X|X_{observed})$ for the given mega-example. To stabilize the metric, we divided the variables into four groups and calculated the average CMLL when observing only one group and hiding the rest. All experiments were conducted on a single machine with 2.4 GHz and 64 GB of RAM.

(Q1, Q2) Friends-and-Smokers MLN: In our first experiment we learned the parameters for the "Friends-and-Smokers" MLN [27], which basically defines rules about the smoking behaviour of people, how the friendship of two people influences whether a person smokes or not, and that a person is more likely to get cancer if he smokes. We enriched the network by adding two clauses: if someone is stressed he is more likely to smoke and people having cancer should get medical treatment. For a given set of parameters we sampled 5 dataset from the joint distribution of the MLN with 10 persons. For each dataset we learned the parameters on this dataset and evaluated on the other four. The ground network of this MLN contains 380 factors and 140 variables. The batchsize was 10 and we used a stepsize of 0.2. Fig. 3(left) shows the CMLL averaged over all of the 5 folds. We ran the lifted piecewiese learning with a batchsize of 10 and a step size of 0.2. Other parameters for SMD were chosen to be $\lambda = .99$, $\mu = 0.1$, and γ the discount for older gradients as 0.9.

As one can see, the lifted SMD has a steep learning curve and has already learned the parameters before seeing the mega example even once (indicated by the black vertical line. Note that we learned the models without stopping criterion and for a fixed number of passes over the data thus the CMLL on the

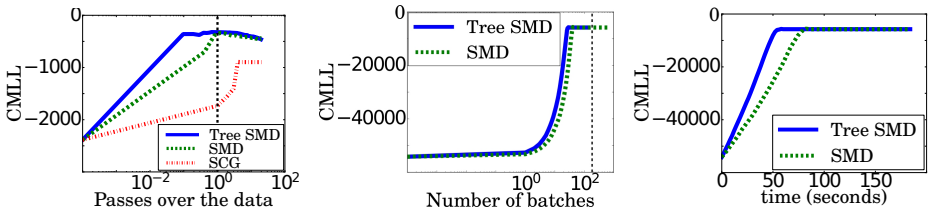


Fig. 4. Experimental results. From left to right, "passes over mega-example" vs. Test-CMLL for the CORA and "number of batches" vs. Test-CMLL for the Wumpus MLNs (the higher the better). The last graph on the right-hand-side shows the runtime vs. CMLL on the Wumpus MLN. As one can see, lifted online learning has already converged before seeing the mega example even once (black vertical line). For the Wumpus MLN, SCG did not converge within 72 hours. (Best viewed in color)

test data can decrease. SCG on the other hand requires four passes over the entire training data to have a similar result in terms of CMLL. Thus **Q1** can be answered affirmatively. Moreover, as Fig 3(center) shows, piecewise learning greatly increases the lifting compared to batch learning, which essentially does not feature lifting at all. Thus, **Q2** can be answered affirmatively.

(Q2,Q3) Voting MLN: To investigate whether tree pieces although more complex can still yield lifting, we considered the Voting MLN from the Alchemy repository. The network contains 3230 factors and 3230 variables. Note that it is a propositional Naive Bayes (NB) model. Hence, depth 0 pieces will yield greater lifting but hamper information flow among attributes if the class variable is unobserved. Tree pieces intuitively couple depth 0 hence will indeed yield lower lifting ratios. However, with larger mini-batches they should still yield higher lifting than the batch case. This is confirmed by the experimental results summarized in Fig 3(right). Thus, **Q3** can be answered affirmatively.

(Q3,Q4) CORA Entity Resolution MLN: In our second experiment we learned the parameters for the Cora entity resolution MLN, one of the standard datasets for relational learning. In the current paper, however, it is used in a non-standard, more challenging setting. For a set of bibliographies the Cora MLN has facts, e.g., about word appearances in the titles and in author names, the venue a paper appeared in, its title, etc. The task is now to infer whether two entries in the bibliography denote the same paper (predicate *samePaper*), two venues are (*sameVenue*), two titles are the same (*sameTitle*), and whether two authors are the same (*sameAuthor*). We sampled 20 bibliographies and extracted all facts corresponding to these bibliography entries. We constructed five folds then trained on four folds and tested on the fifth. We employed a transductive learning setting for this task. The MLN was parsed with all facts for the bibliographies from the five folds, i.e., the queries were hidden for the test fold. The query consisted of all four predicates (*sameAuthor*,*samePaper*,*sameBib*, *sameVenue*). The resulting ground network consisted of 36,390 factors and 11,181 variables. We learnt the parameters using SCG, lifted stochastic meta-descent with standard pieces as well as pieces using relational treefinding with a threshold t of

0.9. The trees consisted of around ten factors on average. So we updated with a batchsize of 100 for the trees and 1000 for standard pieces with a stepsize of 0.05. Furthermore, other parameters were chosen to be $\lambda = .99$, $\mu = 0.9$, and $\gamma = 0.9$. Fig. 4(left) shows the averaged learning results for this entity resolution task. Again, online training does not need to see the whole mega-example; it has learned long before finishing one pass over the entire data. Thus, **(Q4)** can be answered affirmatively.

Moreover, Fig. 4 also shows that by building tree pieces one can considerably speed-up the learning process. They convey a lot of additional information such that one obtains a better solution with a smaller amount of data. This is due to the fact that the Cora dataset contains a lot of strong dependencies which are all broken if we form one piece per factor. The trees on the other hand preserve parts of the local structure which significantly helps during learning. Thus, **(Q3)** can be answered affirmatively.

(Q3,Q4) Lifted Imitation Learning in the Wumpus Domain: To further investigate **(Q3)** and **(Q4)**, we considered imitation learning in a relational domain for a Partially Observed Markov Decision Process (POMDP). We created a simple version of the Wumpus task where the location of Wumpus is partially observed. We used a 5×5 grid with a Wumpus placed in a random location in every training trajectory. The Wumpus is always surrounded by stench on all four sides. We do not have any pits or breezes in our task. The agent can perform 8 possible actions: 4 move actions in each direction and 4 shoot actions in each direction. The agent's task is to move to a cell so that he can fire an arrow to kill the Wumpus. The Wumpus is not observed in all the trajectories although the stench is always observed. Trajectories were created by real human users who play the game. The resulting network contains 182400 factors and 4469 variables. We updated with a batchsize of 200 for the trees and 2000 for standard pieces with a stepsize of 0.05. As for the cora dataset used $\lambda = .99$, $\mu = 0.9$, and $\gamma = 0.9$.

Figure 4 shows the result on this dataset for lifted SMD with standard pieces as well as pieces using relational treefinding with a threshold t of 0.9. For this task, SCG did not converge within 72 hours. Note that this particular network has a complex structure with lots of edges and large clauses. This makes inference on the global model intractable. Fig. 4 (center) shows the learning curve for the total number of batches seen as well as the total time needed for one pass over the data (right). As one can see, tree pieces actually yield faster convergence, again long before having seen the dataset even once. Thus, **(Q3)** and **(Q4)** can be answered affirmatively.

Taking all experimental results together, all questions **Q1**-**Q4** can be clearly answered affirmatively.

6 Conclusions

In this paper, we have introduced the first lifted online training method for relational models. We employed the intuitively appealing idea of separately training

pieces of the full model and combining the results in iteration and turned it into an online stochastic gradient method that processes one lifted piece after the other. We showed that this approach can be justified as maximizing a loose bound on the log likelihood and that it converges to the same quality solution over an order of magnitude faster, simply because unlike batch training it starts optimizing long before having seen the entire mega-example even once.

The stochastic relational gradient framework developed in the present paper puts many interesting research goals into reach. For instance, one should tackle one-pass relational learning by investigating different ways of gain adaption and scheduling of pieces for updates. One should also investigate budget constraints on both the number of examples and the computation time per iteration. In general, relational problems can easily involve models with millions of random variables. At such massive scales, parallel and distributed algorithms for training are essential to achieving reasonable performance.

Acknowledgements. The authors thank the reviewers for their helpful comments. BA and KK were supported by the Fraunhofer ATTRACT fellowship STREAM and by the European Commission under contract number FP7-248258-First-MM. SN gratefully acknowledges the support of the DARPA Machine Reading Program under AFRL prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government.

Bibliography

1. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. The MIT Press (2007)
2. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S.H. (eds.): Probabilistic Inductive Logic Programming. LNCS (LNAI), vol. 4911. Springer, Heidelberg (2008)
3. Singla, P., Domingos, P.: Lifted First-Order Belief Propagation. In: AAAI (2008)
4. Kersting, K., Ahmadi, B., Natarajan, S.: Counting belief propagation. In: UAI, Montreal, Canada (2009)
5. Mihalkova, L., Huynh, T., Mooney, R.: Mapping and revising markov logic networks for transfer learning. In: AAAI, pp. 608–614 (2007)
6. Rosenblatt, F.: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan (1962)
7. Besag, J.: Statistical Analysis of Non-Lattice Data. Journal of the Royal Statistical Society. Series D (The Statistician) 24(3), 179–195 (1975)
8. Winkler, G.: Image Analysis, Random Fields and Dynamic Monte Carlo Methods. Springer (1995)
9. Sutton, C., Mccallum, A.: Piecewise training for structured prediction. Machine Learning 77(2-3), 165–194 (2009)
10. Lee, S.I., Ganapathi, V., Koller, D.: Efficient structure learning of Markov networks using L1-regularization. In: NIPS (2007)
11. Hinton, G.: Training products of experts by minimizing contrastive divergence. Neural Computation 14 (2002)
12. Kok, S., Domingos, P.: Learning Markov logic network structure via hypergraph lifting. In: ICML (2009)

13. Kok, S., Domingos, P.: Learning Markov logic networks using structural motifs. In: ICML (2010)
14. Khot, T., Natarajan, S., Kersting, K., Shavlik, J.: Learning markov logic networks via functional gradient boosting. In: ICDM (2011)
15. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics, 1189–1232 (2001)
16. Natarajan, S., Khot, T., Kersting, K., Guttmann, B., Shavlik, J.: Gradient-based boosting for statistical relational learning: The relational dependency network case. Machine Learning (2012)
17. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
18. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B.39 (1977)
19. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. J. Artif. Intell. Res (JAIR) 15, 391–454 (2001)
20. Kersting, K., De Raedt, L.: Adaptive Bayesian Logic Programs. In: Rouveirol, C., Sebag, M. (eds.) ILP 2001. LNCS (LNAI), vol. 2157, pp. 104–117. Springer, Heidelberg (2001)
21. Getoor, L., Friedman, N., Koller, D., Taskar, B.: Learning probabilistic models of link structure. Journal of Machine Learning Research 3, 679–707 (2002)
22. Thon, I., Landwehr, N., De Raedt, L.: Stochastic relational processes: Efficient inference and applications. Machine Learning 82(2), 239–272 (2011)
23. Gutmann, B., Thon, I., De Raedt, L.: Learning the Parameters of Probabilistic Logic Programs from Interpretations. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part I. LNCS, vol. 6911, pp. 581–596. Springer, Heidelberg (2011)
24. Natarajan, S., Tadepalli, P., Dietterich, T.G., Fern, A.: Learning first-order probabilistic models with combining rules. Annals of Mathematics and AI (2009)
25. Jaeger, M.: Parameter learning for Relational Bayesian networks. In: ICML (2007)
26. Huynh, T., Mooney, R.: Online max-margin weight learning for markov logic networks. In: SDM (2011)
27. Singla, P., Domingos, P.: Lifted first-order belief propagation. In: AAAI (2008)
28. Sutton, C., McCallum, A.: Piecewise training for structured prediction. Machine Learning 77(2-3), 165–194 (2009)
29. Richards, B., Mooney, R.: Learning relations by pathfinding. In: AAAI (1992)
30. Wainwright, M., Jaakkola, T., Willsky, A.: A new class of upper bounds on the log partition function. In: UAI, pp. 536–543 (2002)
31. Schraudolph, N., Graepel, T.: Combining conjugate direction methods with stochastic approximation of gradients. In: AISTATS, pp. 7–13 (2003)
32. Vishwanathan, S.V.N., Schraudolph, N.N., Schmidt, M.W., Murphy, K.P.: Accelerated training of conditional random fields with stochastic gradient methods. In: ICML, pp. 969–976 (2006)
33. Amari, S.: Natural gradient works efficiently in learning. Neural Comput. 10, 251–276 (1998)
34. Le Roux, N., Manzagol, P.A., Bengio, Y.: Topmoumoute online natural gradient algorithm. In: NIPS (2007)
35. Müller, M.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6(4), 525–533 (1993)
36. Ahmadi, B., Kersting, K., Sanner, S.: Multi-Evidence Lifted Message Passing, with Application to PageRank and the Kalman Filter. In: IJCAI (2011)

Scalable Relation Prediction Exploiting Both Intrarelational Correlation and Contextual Information

Xueyan Jiang[2], Volker Tresp[1,2], Yi Huang[1,2], Maximilian Nickel[2], and Hans-Peter Kriegel[2]

[1] Siemens AG, Corporate Technology, Munich, Germany
[2] Ludwig Maximilian University of Munich, Munich, Germany

Abstract. We consider the problem of predicting instantiated binary relations in a multi-relational setting and exploit both intrarelational correlations and contextual information. For the modular combination we discuss simple heuristics, additive models and an approach that can be motivated from a hierarchical Bayesian perspective. In the concrete examples we consider models that exploit contextual information both from the database and from contextual unstructured information, e.g., information extracted from textual documents describing the involved entities. By using low-rank approximations in the context models, the models perform latent semantic analyses and can generalize across specific terms, i.e., the model might use similar latent representations for semantically related terms. All the approaches we are considering have unique solutions. They can exploit sparse matrix algebra and are thus highly scalable and can easily be generalized to new entities. We evaluate the effectiveness of nonlinear interaction terms and reduce the number of terms by applying feature selection. For the optimization of the context model we use an alternating least squares approach. We experimentally analyze scalability. We validate our approach using two synthetic data sets and using two data sets derived from the Linked Open Data (LOD) cloud.

1 Introduction

There recently has been a growing interest in the prediction of the truth values of (instantiated) binary relations, i.e., grounded statements. A major reason is the growing amount of data that is published in the Linked Open Data (LOD) cloud where information is represented in the form of subject-predicate-object (s, p, o) triples. In the associated RDF graph (Resource Description Framework), entities (i.e., subjects and objects) are represented as nodes and statements are represented as directed labeled links from subject node to object node. Thus relation prediction becomes equivalent to the prediction of labeled links. In this paper we focus on the prediction of statements with a common predicate p and with defined sets of subject nodes and object nodes. We then generalize to entities not in the training set. For predicting instantiated binary relations we exploit both

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 601–616, 2012.

intrarelational correlations and contextual information. Intrarelational correlations exploits dependencies within the relation of interest and would correspond to the data dependencies exploited in typical collaborative learning systems. Contextual information consists of all other information sources.

In the concrete examples we consider models that exploit two sources of contextual information. The first one is multi-relational contextual information that is derived from the database. The second one concerns information from unstructured sources, e.g., in form of textual documents describing the involved entities (e.g., from the entities' Wikipedia pages). As a new contribution we exploit nonlinear interactions between the associated information sources. By using low-rank approximations in the context models, the models perform latent semantic analyses and can generalize across specific terms, i.e., the model might use similar latent representations for semantically related terms. In [12] we have introduced a hierarchical Bayesian approach that is highly scalable by exploiting sparse matrix algebra, can easily generalize to new entities and does not suffer from local optima. [13] describes the additive modelling approach in greater detail. In this paper we compare the two approaches and also consider simple heuristic solutions.

The paper is organized as follows. The next section discusses related work. Section 3 describes our different ways of combining contextual information with intrarelational correlations. In section 4 we discuss how context information can be modeled and we introduce an alternating least squares solution for combining intrarelational correlations with contextual information. Section 5 contains our experimental results on synthetic data sets and on two data sets derived from the Linked Open Data (LOD) cloud. We also perform extensive experiments on scalability. Section 6 presents our conclusions.

2 Related Work

Some standard models for relational learning are, e.g., Probabilistic Relational Models [16,9], Markov Logic Networks [24] and the infinite models in [29,15]. Although conceptionally elegant, they are difficult to apply and often involve complex structural learning.[1] Our approach is related to link prediction, which is reviewed in [22,8]. SVD-based decompositions, as used in our approach, were compared to nonnegative matrix factorization (NMF) and latent Dirichlet allocation (LDA) in [10]. All three approaches benefitted greatly from regularization and then gave comparable performance. We used SVD-based decompositions since they can efficiently be computed using highly optimized packages, since predictions for new entities can be calculated easily and since they have unique solutions.

The winning entries in the Netflix competitions are based on matrix factorization [25,1,4]. The main difference is that, in those applications, unknown ratings

[1] As an example, we were not successful in getting the structural learning in MLNs to work in our domains.

can be treated as missing entries. In contrast, in relation prediction an instantiated relationship not known to be true is very likely untrue. In the experiments in our paper we include the hierarchical Bayesian model developed in [12]. An advantage of that model is that it is based on a probabilistic generative model.

RFD graphs also map elegantly to a tensor representation. Tensor models for relational learning have been explored in [20] and [21], showing both scalability and state-of-the-art results on benchmark datasets.

Recently, there has been quite some work on the relationship between kernels and graphs [5,27,7,3,18]. Kernels for semi-supervised learning, for example, have been derived from the spectrum of the Graph-Laplacian. In [30,28] approaches for Gaussian process based link prediction have been presented. Link prediction in relational graphs has also been studied by the relational learning communities and by the ILP communities [26,19,17]. Kernels for semantically rich domains have been developed by [6]. Link prediction is covered and surveyed in [22,8]. Inclusion of ontological prior knowledge to relational learning has been discussed in [23].

3 Relation Prediction by Exploiting Both Intrarelational Correlation and Context Information

3.1 Notation and Contextual Information

In this paper we assume that binary relations are presented by RDF triples of the form (s, p, o) where subject s and object o stand for entities in a domain and where p is the predicate. In an RDF graph, entities are nodes and a triple is a labeled directed link from subject node to object node. Let $Z_{i,j,k}$ be a variable assigned to the triple (s = i, p = j, o = k). $Z_{i,j,k} = 1$ stands for the fact that the corresponding triple is known to exist and $Z_{i,j,k} = 0$ stands for the fact that the corresponding triple is not known to exist.

We are now interested in a particular set of triples $\{(\text{s} = i, \text{p} = p, \text{o} = k)\}_{i,k}$ where p = p is fixed and where the sets of subject and object entities are known. Let X be the matrix of Z-values where $(X)_{i,k} = 1$ if (s = i, p = p, o = k) is known to exist; otherwise $(X)_{i,k} = 0$. In the following we will derive a number of matrices where the zeros are replaced by continuous numbers that can be interpreted as confidence values for a relation being true, based on the available evidence. In a probabilistic sense, we can interpret the continuous numbers as $P((X)_{i,k} = 1|Data)$. These confidence values can then be the basis for classification and ranking tasks as described in Section 5.

In this paper we assume that contextual information is available from which we can derive an estimate of how likely a target relation is true, denoted by

$$f_{i,k}.$$

Contextual information might consist of other statements in the knowledge base relevant for the relation under consideration, but could also include unstructured information, e.g., textual documents describing the involved entities

(see Section 4). The corresponding matrix F with $(F)_{i,k} = f_{i,k}$ has the same dimensionality as X and entries typically assume values between zero and one, although this is not enforced.

Our first and most simple estimate for the confidence values for statements would be simply derived from this estimate and represents a pure context-based model of the form

$$X_{\mathrm{F}} = F. \tag{1}$$

Here we do not exploit intrarelational correlations and this solution is sensible if X is very sparse. Alternatively, we might trust the ones in the X matrix (which represent certain facts) and use

$$X_{\mathrm{M}} = \max(X, F) \tag{2}$$

where max is applied componentwise or, if we are willing to tolerate confidence values greater than one,

$$X_{\mathrm{S}} = X + F. \tag{3}$$

In both solutions, F is mostly relevant to the zero entries of X.

3.2 Intrarelational Correlations

In many applications the correlations in the relational matrix can be exploited to derive predictions, an effect often associated with collaborative filtering. The leading approaches exploiting intrarelational correlations are based on a factorization of X, which is also the approach we are taking.

We propose to minimize the cost function

$$\min_{X_{\mathrm{r}}} \|X - X_{\mathrm{r}}\|_F^2$$

where we impose the constraint on X_{r} to have a maximum rank of r.

It is well known that one specific solution can be derived from a singular value decomposition (SVD) with

$$X = UDV^T \tag{4}$$

where U and V are matrices with orthonormal columns and where D is a diagonal matrix. The diagonal entries $d_i \geq 0$ are ordered according to magnitude. The optimal r-rank reconstruction can be written as

$$X_{\mathrm{r}} = U_r D_r \, V_r^T$$

where U_r and V_r contain the first r columns of the respective matrices and where D_r is a diagonal matrix with the r leading components of D. Low-rank reconstructions are used in latent semantic analysis to generalize from observed terms to related terms, they are an important ingredient in the winning entries in the Netflix competition [25,1,4], and they also give very good results in predicting links in semantic graphs [10].

We also consider a regularized version which typically improves predictions significantly by using the cost function

$$\min_{\tilde{W}} \left(\|X - \tilde{W} X_{\mathrm{r}}\|_F^2 + \lambda \|\tilde{W}\|_F^2 \right)$$

where X_{r} is fixed and where the parameter matrix $\tilde{W}$ is optimized.

The overall solution is then[2]

$$X_{\mathrm{CF}} = U_r \operatorname{diag}\left\{ \frac{d_i^3}{d_i^2 + \lambda} \right\}_{i=1}^{r} V_r^T$$

$$= U_r \operatorname{diag}\left\{ \frac{d_i^2}{d_i^2 + \lambda} \right\}_{i=1}^{r} U_r^T X = X V_r \operatorname{diag}\left\{ \frac{d_i^2}{d_i^2 + \lambda} \right\}_{i=1}^{r} V_r^T \tag{5}$$

where $\operatorname{diag}\left\{ \frac{d_i^2}{d_i^2+\lambda} \right\}_{i=1}^{r}$ is an $r \times r$ diagonal matrix with r diagonal entries. In the following we assume that X has fewer rows than columns such that U_r is fast to compute based on an SVD of the kernel matrix XX^T, but one should simply apply the reconstruction most suitable.

We can easily generalize to a new subject entity with x^{new} (as column vector) using

$$x_{\mathrm{CF}}^{new} = V_r \operatorname{diag}\left\{ \frac{d_i^2}{d_i^2 + \lambda} \right\}_{i=1}^{r} V_r^T x^{new} = X^T U_r \operatorname{diag}\left\{ \frac{1}{d_i^2 + \lambda} \right\}_{i=1}^{r} U_r^T X x^{new}. \tag{6}$$

We can now include contextual information by adding the context matrix and the intrarelational module and obtain as a heuristics

$$X_{\mathrm{H}} = X_{\mathrm{CF}} + F. \tag{7}$$

Note that in contrast to X_{S}, here we use X_{CF} instead of X and we obtain a combination model that exploits correlations in X. Thus we will get a high score for a link, if either the context model or the intrarelational model (or both) is positive about the link.

3.3 Hierarchical Bayes

So far, the combination scheme in Equation 7 might be considered a plausible heuristic. In this section and in the next section we consider two combination schemes that can be derived from principled approaches.

In [12] we described a hierarchical Bayesian (HB) approach for the combination of contextual information with intrarelational correlation. It motivates the following approach: We are searching for the low-rank approximation X_{HBS} that minimizes

$$\min_{X_{\mathrm{HBS}}} \|X_S - X_{\mathrm{HBS}}\|_F^2$$

[2] Here and in the following we have typically these three ways of formulating the solution. One should take the one which is most efficient considering the dimensionalities of the involved matrices.

where $X_{\mathrm{S}} = X + F$ was defined in Equation 3. Again, the solution can be based on the SVD, in this case in the form of

$$X + F = U^{MF} D^{MF} V^{MF^T}$$

and a regularized low-rank approach now leads to the model

$$X_{\mathrm{HBS}} = U_r^{MF} \operatorname{diag}\left\{ \frac{(d_i^{MF})^2}{(d_i^{MF})^2 + \lambda} \right\}_{i=1}^{r} U_r^{MF^T}(X + F). \tag{8}$$

A similar solution, i.e., X_{HBM}, is obtained if we use X_{M} instead of X_{S}. Note that for X_H we first smooth X via a regularized low-rank approximation and then add F, whereas for M_{HBS} we first add X and F and then smooth the resulting matrix.

Alternatively we can use as a basis the decomposition of X instead of the decomposition of $X + F$ and obtain

$$\begin{aligned} X_{\mathrm{HBS2}} &= U_r \operatorname{diag}\left\{ \frac{d_i^2}{d_i^2 + \lambda} \right\}_{i=1}^{r} U_r^T(X + F) \\ &= X_{\mathrm{CF}} + U_r \operatorname{diag}\left\{ \frac{d_i^2}{d_i^2 + \lambda} \right\}_{i=1}^{r} U_r^T F. \end{aligned} \tag{9}$$

For X_{HBS2} we can exploit sparse matrix algebra for calculating the decomposition of X (whereas $X + F$ is typically non sparse) and F only needs to be calculated for the entities of interest. Interestingly, the solution consists of adding to X_{CF} a regularized projections of F using the *largest* singular values, so we add to X_{CF} a "low-frequency" version of F.

3.4 Additive Models

The idea here is that the intrarelational correlations should only model the residual difference after F has been subtracted from X. The goal is then to minimize the cost function

$$\min_{X_{\mathrm{CFa}}} \|X - (F + X_{\mathrm{CFa}})\|_F^2.$$

A regularized low-rank approximation where the basis is calculated from the decomposition of X is then

$$X_{\mathrm{CFa}} = U_r \operatorname{diag}\left\{ \frac{d_i^2}{d_i^2 + \lambda} \right\}_{i=1}^{r} U_r^T(X - F) \tag{10}$$

and the overall prediction is

$$X_{\mathrm{add}} = X_{\mathrm{CFa}} + F$$

such that

$$X_{\text{add}} = X_{\text{CFa}} + F = U_r \operatorname{diag}\left\{\frac{d_i^2}{d_i^2+\lambda}\right\}_{i=1}^{r} U_r^T(X-F) + F$$

$$= X_{\text{CF}} + U\left(I - \operatorname{diag}\left\{\frac{d_i^2}{d_i^2+\lambda}\right\}_{i=1}^{r}\right) U^T F. \tag{11}$$

Interestingly, the solution consists of adding to X_{CF} a "high frequency" version of F.

In the next section we derive specific models for F. An overall additive model where F and X_{add} are adapted in turn (the latter using Equation 10) and where Equation 11 is used for overall prediction is defined as X_{global}.

4 Context Models for Our Applications

4.1 Context Models Based on the Database

So far, f could have been an arbitrary function of context information. We see this as a great advantage of our approach since it permits a great modularity and the context model and the intrarelational model can be optimized independently.

Now we derive a specific context model that we will use in the applications. Let's consider a multi-relational database of triples (i.e., a triple store). Let A be a matrix with as many rows as X, i.e., with one row for each subject entity in X. The columns of A represent features describing the subjects in X. In the simplest case they consist of the truth value of all (relevant) triples with the same subject. Consider the example that rows are users and columns are movies and the task is to predict if a user watches a movie. In this example, a particular column in A might indicate if a user is of young age and the model would be able to exploit the preference of young people for certain movies.

Similarly, B is a matrix. The number of rows of B is equal to the number of columns of X.The columns of B represent features describing the objects in X. In the simplest case they consist of the truth values of all (relevant) triples, where the object of X is the subject. Following the example, a column in B might indicate if a movie is an action movie and the model can exploit the preference of some people for action movies. Thus B is suitable to model personal preferences.

Finally, we introduce the matrix C formed by the Kronecker product $C = A \otimes B$, i.e., C contains all possible product terms of the elements of A and B. The number of rows in C is the number of rows of A times the number of rows of B and the number of columns in C is the number of columns of A times the number of columns of B. Following the example, a column in C might indicate if a movie is an action movie and, at the same time, the user is young and the model might learn that young people like action movies.

We now write a least squares cost function

$$\|X - F\|_F^2$$

where

$$F = AW^A + (BW^B)^T + \text{matrix}(Cw^C) \tag{12}$$

and where matrix$(\cdot)$ transforms the vector into a matrix of appropriate dimensions. $\|\cdot\|_F$ is the Frobenius norm. The matrices W^A and W^B and the vector w^C contain the parameters to be optimized. Thus we predict the entries in F as a linear combination of the subject features in A, the column features in B and the interaction features in C.

To control overfitting, we add to the cost functions the penalty terms $\lambda_A\|W^A\|_F^2$, $\lambda_B\|W^B\|_F^2$, and $\lambda_C\|w^C\|_F^2$.

To reduce the amount of computation and also as a means to prevent overfitting, we are looking for low-rank solutions with ranks r_A, r_B, and r_C as discussed in the next subsection. By using low-rank models, the models perform latent semantic analyses and can generalize across specific terms, i.e., the model might use similar latent representation for semantically related terms.

The number of interaction terms in C can easily be several millions, so we perform fast feature selection strategies by evaluating the Pearson correlation between targets and features.

4.2 Alternating Least Squares

An easy way to optimize the cost function is to repeatedly iterate over all three terms where in each iteration we keep the other two fixed.

Let

$$\begin{aligned} X^{-A} &= X - \left((BW^B)^T + \text{matrix}(Cw^C)\right) \\ X^{-B} &= X - \left(AW^A + \text{matrix}(Cw^C)\right) \\ X^{-C} &= X - \left(AW^A + (BW^B)^T\right) \end{aligned}$$

and let $x^{-C} = \text{vec}(X^{-C})$.

The individuals contributions are the calculated as

$$AW^A = U_{A,r_A}\ \text{diag}\left\{\frac{(d_i^{(A)})^2}{(d_i^{(A)})^2 + \lambda_A}\right\}_{i=1}^{r_A} U_{A,r_A}^T X^{(-A)} \tag{13}$$

$$BW^B = U_{B,r_B}\ \text{diag}\left\{\frac{(d_i^{(B)})^2}{(d_i^{(B)})^2 + \lambda_B}\right\}_{i=1}^{r_B} U_{B,r_B}^T (X^{(-B)})^T \tag{14}$$

$$Cw^C = U_{C,r_C}\ \text{diag}\left\{\frac{(d_i^{(C)})^2}{(d_i^{(C)})^2 + \lambda_C}\right\}_{i=1}^{r_C} U_{C,r_C}^T x^{(-C)} \tag{15}$$

where we have used the singular value decompositions (SVD)

$$A = U_A D_A V_A^T \quad B = U_B D_B V_B^T \quad C = U_C D_C V_C^T \tag{16}$$

and where U_{A,r_A} contains the first r_A columns of U_A, U_{B,r_B} contains the first r_B columns of U_B, and U_{C,r_C} contains the first r_C columns of U_C.

Again we can exploit sparse matrix algebra for calculating the decompositions. The convergence of the alternating least squares algorithm is quite fast, requiring fewer than 10 iterations.

Note again that we can include the intrarelational model as an additional fourth component to be optimized with alternating least squares leading to the model X_{global} introduced in Section 3.4. A more extensive analysis of the additive models can be found in [13] where also additional feature candidates are discussed. Since the bases for the decompositions (calculated in Equations 4 and 16) are calculated before the optimization of the parameters, the alternating least squares iterations converge to unique solutions. [3]

4.3 Incorporating External Information Sources and Aggregation

In the applications we are considering we sometimes have available textual data describing the involved entities. We simply treat the keywords in the textual descriptions as additional features describing subjects, resp. objects. In some applications, it is useful to add aggregated information. This can be represented as additional features as well.

5 Experiments

5.1 Scalability

For the kind of relational data that we are considering, X is very sparse and the reduced-rank reconstruction can be calculated efficiently. Figure 1 shows experimental results. Note that for a sizable X-matrix with 10^5 rows, 10^6 columns, 10^7 nonzero elements and a rank of $r = 50$, the computation only takes approximately 10 minutes on a standard laptop computer. For matrices where $K = XX^T$ becomes dense one might employ the alternating least squares solution described in [20] that does not rely on a sparsity of K in the factorization and does not enforce orthogonality constraints.

5.2 Tuning of Hyperparameters

The approaches contain up to 8 hyperparameters (r, r_A,r_B,r_C, λ, λ_A,λ_B,λ_C) which are tuned using cross-validation sets (i.e. they are not tuned on the test set). We follow the approach described in [2] and perform a random search for the best hyperparameters.

[3] Recall that we first calculate the kernel matrix $K = XX^T$ and then perform the SVD decomposition. Naturally, we could start with a kernel matrix suitable for the RDF graph. In this view our alternating least squares solution is an efficient way of calculating a kernel solution with a kernel $k(s, s', o, o') = k_{CF}(s, s') + k_A(s, s') + k_B(o, o') + k_C(s, s', o, o')$ where $k_{CF}(s, s')$ is the intrarelational kernel, $k_A(s, s')$ is a kernel for subject nodes, $k_B(o, o')$ is a kernel for object nodes, and $k_C(s, s', o, o')$ is a kernel for modeling interactions.

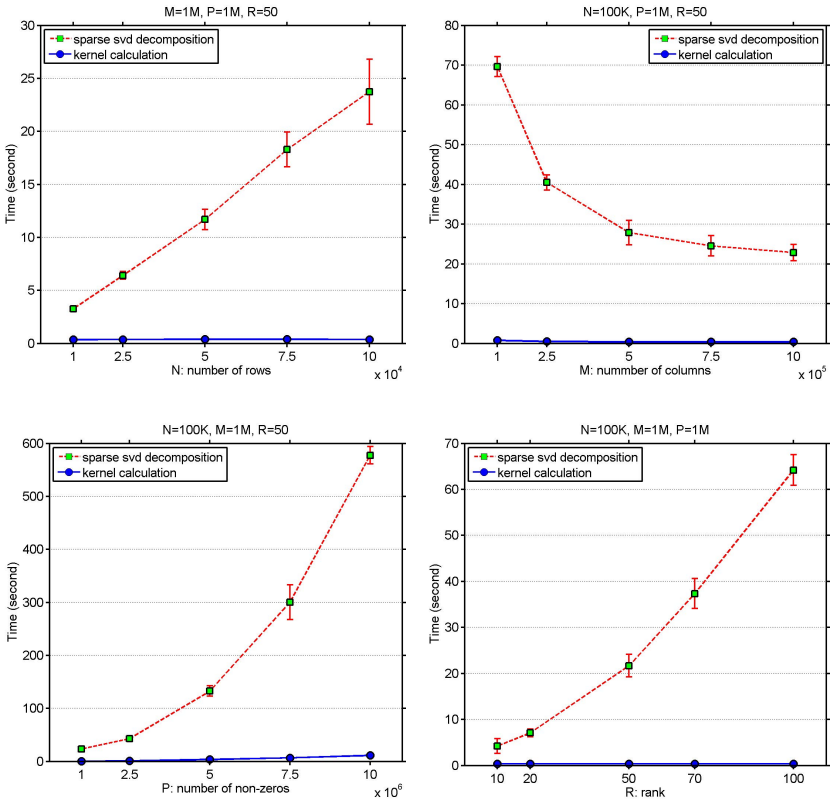


Fig. 1. We consider a sparse random $N \times M$ matrix X. First we construct the kernel matrix via $K = XX^T$ and then use sparse SVD to obtain U_r. The top left figure shows computational time for the SVD as a function of N (red dashed). We see approximately a linear dependency which is related to the fact that the number of rows of U is N as well. In this experiment, $r = 50$, $M = 10^6$ and the number of nonzero entries in X is $p = 10^6$. The top right figure shows computation time for the SVD as a function of M (red dashed). We see a decrease: the reason is that with increasing M, K becomes less dense. We used $p = 10^6$, $N = 10^5$, and $r = 50$. The bottom left shows an approximately quadratic dependency of the computational time for the SVD on p ($M = 10^6$, $N = 10^5$, $r = 50$) (red dashed). Note that the last data point in the plot is a system with $p = 10^7$ requiring only 10 minutes of computation. Finally, the bottom right figure shows the dependency on r ($M = 10^6$, $N = 10^5$, $p = 10^6$) (red dashed). A 10 fold increase in r approximately displays a 10 fold increase in computational cost. Each figure also shows the computational time for calculating $K = XX^T$, which, in comparison, is negligible (blue continuous). A prediction for data for a novel subject (i.e., a new row in X) can efficiently be calculated using Equation 6.

5.3 Synthetic Data

The synthetic data has been generated according to our modeling assumptions. The target relation is a sum of four components: the first one is modeling the intrarelations correlation, the second one uses features describing the subject entities, the third one uses features describing the object entities, and the fourth one uses interaction terms. In the first experiment, both the intrarelational correlation and the context models have predictive power and all six combination schemes improve upon the subsystems. The additive models X_{add} and X_{global} seem to be more robust and perform well on both experiments.

We randomly selected one true relation to be treated as unknown (test statement) for each subject entity in the data set. In the test phase we then predicted all unknown relations for the entity, including the entry for the test statement. The test statement should obtain a high likelihood value, if compared to the other unknown entries. The normalized discounted cumulative gain (nDCG@all) [11] is a measure to evaluate a predicted ranking.

5.4 Associating Diseases with Genes

As the costs for gene sequencing are dropping, it is expected to become part of clinical practice. Unfortunately, for many years to come the relationships between genes and diseases will remain only partially known. The task here is to predict diseases that are likely associated with a gene based on knowledge about gene and disease attributes and about known gene-disease patterns.

Disease genes are those genes involved in the causation of, or associated with a particular disease. At this stage, more than 2500 disease genes have been discovered. Unfortunately, the relationship between genes and diseases is far from simple since most diseases are polygenic and exhibit different clinical phenotypes. High-throughput genome-wide studies like linkage analysis and gene expression profiling typically result in hundreds of potential candidate genes and it is still a challenge to identify the disease genes among them. One reason is that genes can often perform several functions and a mutational analysis of a particular gene reveals dozens of mutation cites that lead to different phenotype associations to diseases like cancer [14]. An analysis is further complicated since environmental and physiological factors come into play as well as exogenous agents like viruses and bacteria.

Despite this complexity, it is quite important to be able to rank genes in terms of their predicted relevance for a given disease as a valuable tool for researchers and with applications in medical diagnosis, prognosis, and a personalized treatment of diseases.

In our experiments we extracted information on known relationships between genes and diseases from the LOD cloud, in particular from Linked Life Data and Bio2RDF, forming the triples (Gene, related_to, Disease). In total, we considered 2462 genes and 331 diseases. For genes we extracted 11332 features and for the diseases 1283 features from the LOD cloud. In addition, we retrieved 8000 textual features describing genes and 3800 textual features describing diseases

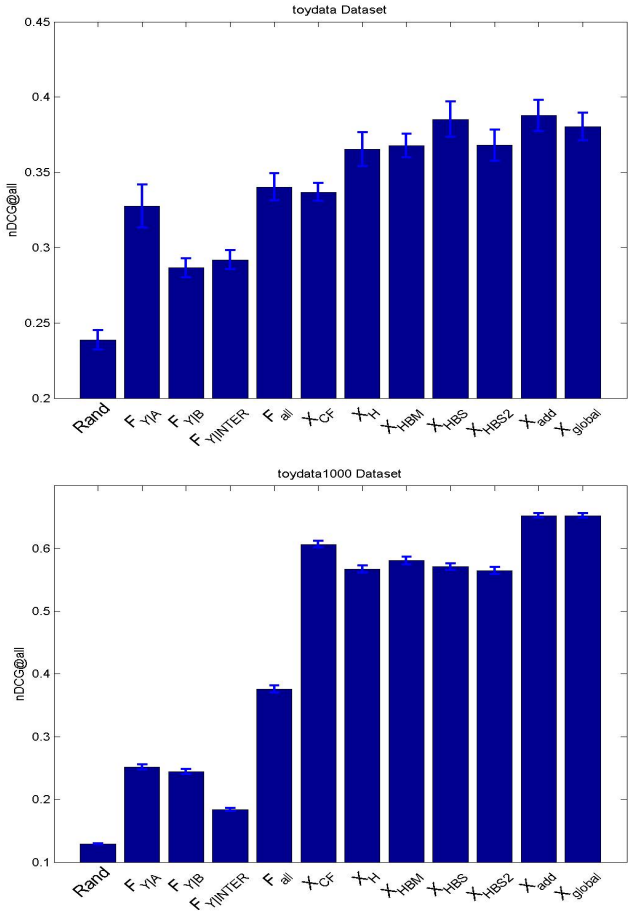


Fig. 2. Test results on synthetic data. In the first experiment (top), we had 100 subjects and 80 objects, A had 80 columns, B had 80 columns, and C had 8000 rows and 4000 columns. The three context models $F_{Y|A}$, $F_{Y|B}$, and $F_{Y|INTER}$ make valuable predictions significantly above random. The combination of all three context models, i.e., $X_{\mathrm{F}} = F_{\mathrm{all}}$, is better than any of the individual context models. The context model and the intrarelation correlation are comparable strong in prediction: X_{CF} gives comparable results to F_{all}. All six combination schemes are better than the intrarelational model or the context model on their own, so all combination schemes are sensible. The additive model X_{add} and the additive model where the context model and the context model are jointly optimized (X_{global}) perform best, although there is no statistical significant difference between the 6 combination models. In the second experiment (bottom), we had 1000 subjects and 1000 objects, A had 6 columns, B had 7 columns, and C had 1000000 rows and 42 columns. Thus the intrarelation correlation is stronger than the contextual model. X_{add} and X_{global} show better performance than X_{CF} and F_{all} individually.

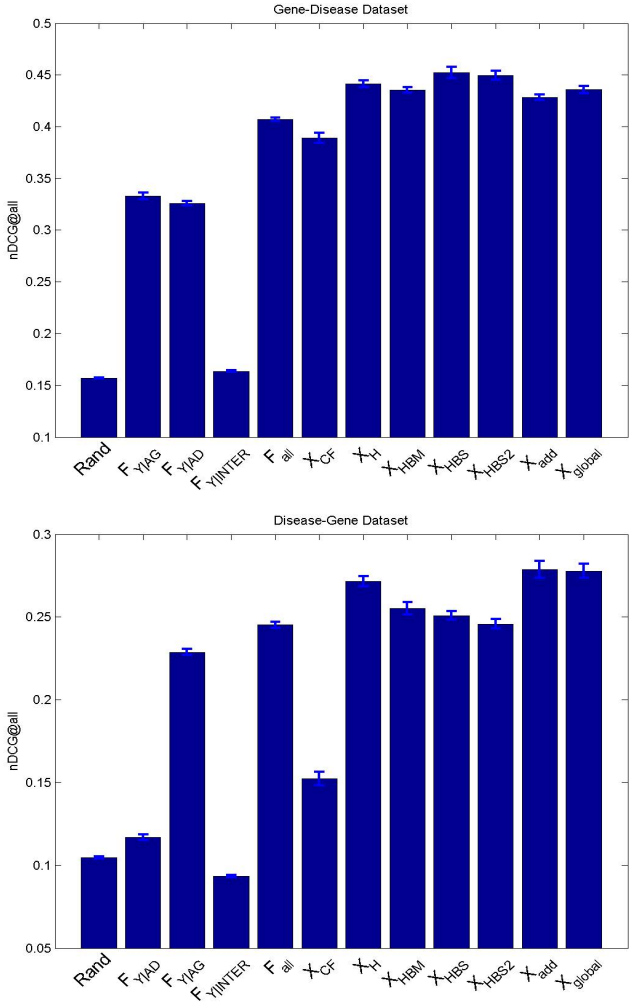


Fig. 3. The goal is to predict the relationship between genes and diseases. On the top we ranked recommended diseases for genes and on the bottom we ranked recommended genes for diseases. We considered contextual features from disease attributes $F_{Y|AD}$ and from gene attributes $F_{Y|AG}$, and contribution from the interaction term $F_{Y|INTER}$. The combination of all contextual models in F_{all} is better than the individual context models where $F_{Y|INTER}$ is not better than random. All six combination schemes are better than the intrarelational model or the context model on their own, so all combination schemes are sensible. In this experiment, two of the hierarchical Bayes models, i.e., X_{HBS} and X_{HBS2} give best results. The results are generally better than the results reported in [12] since, there, only contextual features from text documents were used. The second task, predicting genes for diseases, is more difficult due to the great number of potential genes. Intrarelational correlation on its own is relatively weak (X_{CF}). Again, all combination schemes give good results.

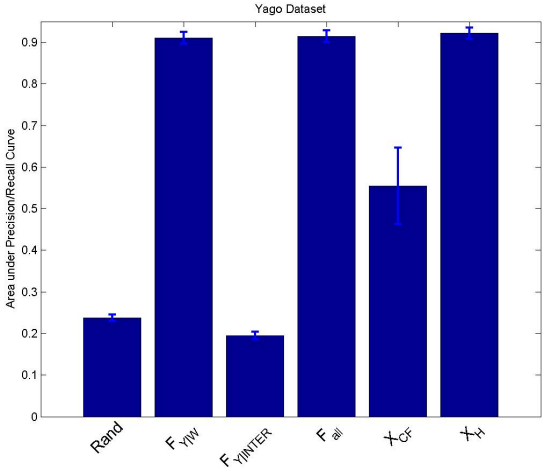


Fig. 4. The task is to predict the nationalities of writers. The writer attributes $F_{Y|W}$ have considerable predictive power. The intrarelational correlation (X_{CF}) benefits from the imbalance of classes. We display the area under precision/recall curve on writers not in the training set (induction). None of the combination models is significantly better than $F_{Y|W}$, which in this experiment is reasonable, since very few writers have more than one nationality (of the combination schemes, we only show X_{H}).

from corresponding text fields in Linked Life Data and Bio2RDF. After applying feature selection, the interaction matrix C had 814922 rows and 1133 columns.

Figure 3 shows the results. This is a very interesting data set: when predicting diseases for genes, the contextual information (reflected in F_{all}) and the intrarelation correlational (reflected in X_{CF}) are both equally strong; in most data sets, one of the two is dominating. All six combination schemes are effective and provide results significantly better than F_{all} or X_{CF} on their own. Predicting genes for diseases generally gives a weaker nDCG score and the leading approaches are X_{add} and X_{global}.

5.5 Predicting Writer's Nationality in YAGO2

The final set of experiments was done on the YAGO2 semantic knowledge base. YAGO2 is derived from Wikipedia and also incorporates WordNet and GeoNames. There are two available versions of YAGO2: core and full. We used the first one which currently contains 2.6 million entities, and describes 33 million facts about these entities. Our experiment was designed to predict the nationalities of writers. We choose four different types of writers: American, French, German and Japanese.

We obtained 440 entities representing the selected writers. We selected 354 entities (i.e., writers) and added textual information describing the writers and the countries. We performed 10-fold cross validation for each model, and

evaluated them with the area under precision and recall curve. Figure 4 shows the results. As there are only 4 nationalities, which are almost always mutual exclusive (there is a small number of writers with more than one nationality), the intrarelational correlation is quite weak and the country attributes were not used. Interestingly, the interaction term is reasonable strong ($F_{\mathrm{Y|INTER}}$). In fact, no model is better than $F_{Y|W}$ which only exploits the contextual information of the writers.

6 Conclusions

In this paper we have considered the problem of predicting instantiated binary relations in a multi-relational setting and exploit both intrarelational correlations and contextual information. We have presented a number of sensible algorithms. The algorithms are all modular and have unique solutions. As contextual information we consider information extracted from the database and textual data describing the entities. To include contextual information we use an alternating least squares approach that includes models for subject features, object features and an interaction model. By using low-rank approximations in the context models, the models perform latent semantic analyses and can generalize across specific terms, i.e., the model might use similar latent representation for semantically related terms. The approaches can exploit sparse matrix algebra and, as we have demonstrated experimentally, are highly scalable. The models can easily be applied to new entities not considered in model training. We presented experimental results on synthetic data, on life science data from the Linked Open Data (LOD) cloud. All the presented combination schemes are effective and there is no clear best approach, although there seems to be a general advantage for the additive models X_{add} and X_{global}.

References

1. Bell, R.M., Koren, Y., Volinsky, C.: All together now: A perspective on the netflix prize. Chance (2010)
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research (2012)
3. Bloehdorn, S., Sure, Y.: Kernel methods for mining instance data in ontologies. In: ESWC (2007)
4. Candes, E.J., Recht, B.: Exact matrix completion via convex optimization. Computing Research Repository - CORR (2008)
5. Cumby, C.M., Roth, D.: On kernel methods for relational learning. In: ICML (2003)
6. D'Amato, C., Fanizzi, N., Esposito, F.: Non-parametric statistical learning methods for inductive classifiers in semantic knowledge bases. In: IEEE International Conference on Semantic Computing, ICSC (2008)
7. Gärtner, T., Lloyd, J.W., Flach, P.A.: Kernels and distances for structured data. Machine Learning (2004)
8. Getoor, L., Diehl, C.P.: Link mining: a survey. SIGKDD Explorations (2005)

9. Getoor, L., Friedman, N., Koller, D., Pfeffer, A., Taskar, B.: Probabilistic relational models. In: Introduction to Statistical Relational Learning (2007)
10. Huang, Y., Tresp, V., Bundschus, M., Rettinger, A., Kriegel, H.-P.: Multivariate Prediction for Learning on the Semantic Web. In: Frasconi, P., Lisi, F.A. (eds.) ILP 2010. LNCS, vol. 6489, pp. 92–104. Springer, Heidelberg (2011)
11. Järvelin, K., Kekäläinen, J.: IR evaluation methods for retrieving highly relevant documents. In: SIGIR 2000 (2000)
12. Jiang, X., Huang, Y., Nickel, M., Tresp, V.: Combining Information Extraction, Deductive Reasoning and Machine Learning for Relation Prediction. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 164–178. Springer, Heidelberg (2012)
13. Jiang, X., Tresp, V., Huang, Y., Nickel, M., Kriegel, H.-P.: Link Prediction in Multi-relational Graphs using Additive Models (submitted, 2012)
14. Kann, M.G.: Advances in translational bioinformatics: computational approaches for the hunting of disease genes. In: Briefings in Bioinformatics (2010)
15. Kemp, C., Tenenbaum, J.B., Griffiths, T.L., Yamada, T., Ueda, N.: Learning systems of concepts with an infinite relational model. In: AAAI (2006)
16. Koller, D., Pfeffer, A.: Probabilistic frame-based systems. In: AAAI (1998)
17. Landwehr, N., Passerini, A., De Raedt, L., Frasconi, P.: kFOIL: Learning simple relational kernels. In: AAAI (2006)
18. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph Kernels for RDF Data. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 134–148. Springer, Heidelberg (2012)
19. Muggleton, S.H., Lodhi, H., Amini, A., Sternberg, M.J.E.: Support Vector Inductive Logic Programming. In: Hoffmann, A., Motoda, H., Scheffer, T. (eds.) DS 2005. LNCS (LNAI), vol. 3735, pp. 163–175. Springer, Heidelberg (2005)
20. Nickel, M., Tresp, V., Kriegel, H.-P.: A three-way model for collective learning on multi-relational data. In: ICML (2011)
21. Nickel, M., Tresp, V., Kriegel, H.-P.: Factorizing YAGO: scalable machine learning for linked data. In: WWW (2012)
22. Popescul, A., Ungar, L.H.: Statistical relational learning for link prediction. In: Workshop on Learning Statistical Models from Relational Data (2003)
23. Rettinger, A., Nickles, M., Tresp, V.: Statistical Relational Learning with Formal Ontologies. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 286–301. Springer, Heidelberg (2009)
24. Richardson, M., Domingos, P.: Markov logic networks. In: Machine Learning (2006)
25. Takacs, G., Pilaszy, I., Nemeth, B., Tikk, D.: On the gravity recommendation system. In: Proceedings of KDD Cup 2007 (2007)
26. Taskar, B., Wong, M.F., Abbeel, P., Koller, D.: Link prediction in relational data. In: NIPS (2003)
27. Vishwanathan, S.V.N., Schraudolph, N., Kondor, R.I., Borgwardt, K.: Graph kernels. Journal of Machine Learning Research - JMLR (2008)
28. Xu, Z., Kersting, K., Tresp, V.: Multi-relational learning with gaussian processes. In: IJCAI (2009)
29. Xu, Z., Tresp, V., Yu, K., Kriegel, H.-P.: Infinite hidden relational models. In: UAI (2006)
30. Yu, K., Chu, W., Yu, S., Tresp, V., Xu, Z.: Stochastic relational models for discriminative link prediction. In: NIPS (2006)

Relational Differential Prediction

Houssam Nassif[1], Vítor Santos Costa[2],
Elizabeth S. Burnside[1], and David Page[1]

[1] University of Wisconsin, Madison, USA
[2] University of Porto, Portugal

Abstract. A typical classification problem involves building a model to correctly segregate instances of two or more classes. Such a model exhibits differential prediction with respect to given data subsets when its performance is significantly different over these subsets. Driven by a mammography application, we aim at learning rules that predict breast cancer stage while maximizing differential prediction over age-stratified data. In this work, we present the first multi-relational differential prediction (aka uplift modeling) system, and propose three different approaches to learn differential predictive rules within the Inductive Logic Programming framework. We first test and validate our methods on synthetic data, then apply them on a mammography dataset for breast cancer stage differential prediction rule discovery. We mine a novel rule linking calcification to *in situ* breast cancer in older women.

Keywords: Uplift modeling, relational data mining, differential prediction, inductive logic programming, ILP, stratified data, breast cancer, in situ.

1 Introduction

A recurrent problem in social sciences is to understand why two or more different populations exhibit differences in a trait. In psychology [8,20,36], one may want to assess the fairness of a test over several different populations. In marketing [17,27,21], one may want to compare subjects and controls in order to study the effectiveness of an advertising campaign. Similar tasks thus arise in several domains and depending on the domain, the problem is known as *differential prediction*, *differential response analysis*, or *uplift modeling*.

In contrast to most studies of *differential prediction* in psychology, marketing's *uplift modeling* assumes an active agent. But, given that in both cases we have two populations that have been subjected to an external agent, we argue that the concepts and techniques originally developed for uplift marketing can, and should, apply to the task of differential prediction (and vice versa). Differential prediction has been studied extensively in the context of multi-attribute data [30,28]. One approach is to generate different classifiers for each sub-population, and to look for differences between the classifiers. Further progress requires building models driven by evaluation functions that take into

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 617–632, 2012.

account the differential nature of uplift modeling [29]. Also, techniques such as *uplift curves* have made it possible to evaluate and compare differential models.

An important differential problem arises in the area of breast cancer research. Breast cancer is the most common type of cancer among women, with a 12% probability of incidence in a lifetime [3]. Breast cancer has two basic stages: an earlier *in situ* stage where cancer cells are still confined where they developed, and a subsequent *invasive* stage where cancer cells infiltrate surrounding tissue. Since nearly all in situ cases can be cured [2], current practice is to treat in situ occurrences in order to avoid progression into invasive tumors [3]. Nevertheless, the time required for an in situ tumor to reach invasive stage may be sufficiently long for a woman to die of other causes; raising the possibility that the diagnosis and treatment may not have been necessary, a phenomenon called *overdiagnosis*.

Cancer occurrence and stage are determined through biopsy, a costly, invasive, and potentially painful procedure. Actual treatment is costly, and may generate undesirable side-effects. For these reasons, the 2009 US National Institutes of Health consensus conference on ductal carcinoma in situ highlighted the need for methods that can accurately identify patient subgroups that would benefit most from treatment, as well as those who do not need treatment [1]. In recent work, Nassif *et al.* [25] reported that different pre-biopsy mammographic features can indeed be used to classify cancer as invasive or in situ *for different age groups*. They identified invasive/in situ classification rules that have significantly different performance across age strata. This finding confirms that, based on age, different mammographic features can be used to classify cancer stage. The key motivation to this work is *to understand how breast cancer evolves differently across different age groups, and what features exhibit differential cancer stage prediction across age*.

Differential breast cancer prediction introduces two novel problems to differential prediction. First, in order to classify a sample, best results require taking into account previous and simultaneous samples for the same patient [9]. This demands a multi-relational data representation. We thus need a *relational* differential model. Second, it is of utmost importance that experts be able to interpret the results and identify patient subgroups. Both challenges can be addressed by using *rules* to represent the model.

We hereby introduce a rule-based multi-relational differential classifier, and demonstrate its applicability on medical data. This work makes three main contributions. First, we present the first multi-relational differential modeling system, and introduce, implement and evaluate novel methods to guide search in a rule-based differential setting. We propose three general methods that are implemented within the Inductive Logic Programming (ILP) framework [23,11], a commonly used approach for relational data mining. We opt for ILP-based rule learning instead of decision-tree-based rule learning because the latter is a special case of the former [6,34]. Second, we present a detailed evaluation of the applicability and usefulness of our approach under different data sizes and noise rates through simulated data. Third, we demonstrate that the system can indeed obtain differential rules of interest to an expert on real data.

2 Related Work

To the best of our knowledge, differential prediction was first used in psychology to assess the fairness of cognitive and educational tests. In this area, it is defined as the case where consistent nonzero errors of prediction are made for members of a given subgroup [8], and it is detected by fitting a common regression equation and checking for systematic prediction discrepancies for given subgroups, or by building regression models for each subgroup and testing for differences between the resulting models [20,36]. The standard approach uses moderated multiple regression, where the criterion measure is regressed on the predictor score, subgroup membership, and an interaction term between the two [5,33]. If the predictive model differs in terms of slopes or intercepts, it implies that bias exists because systematic errors of prediction would be made on the basis of group membership.

An example is assessing how college admission test scores predict first year cumulative grades for males and females. For each gender group, we fit a regression model. We then compare the slope, intercept and/or standard errors for both models. If they differ, then the test exhibits differential prediction and may be considered unfair.

The same concept arises in case-control studies, and is referred to as *differential misclassification*. Instances are cross-classified by case-control status and exposure category. An exposure misclassification is defined as differential if the probabilities of misclassification differ for instances with different case-control categories. Similarly, a case-control misclassification is defined as differential if the probabilities of misclassification differ for instances with different exposure categories [7,13]. This concept is the basis of the related machine learning concept of "differential misclassification cost", incorporating different misclassification costs into a cost sensitive classifier [31].

An important application of differential prediction is in marketing studies, where it can be used to understand the best targets for an advertising campaign and it is often known as uplift modeling. Seminal work includes Radcliffe and Surry's true response modeling [27], Lo's true lift model [21], and Hansotia and Rukstales' incremental value modeling [17]. As an example, Hansotia and Rukstales construct a regression and a decision tree, or CHART, model to identify customers for whom direct marketing has sufficiently large impact. The splitting criterion is obtained by computing the difference between the estimated probability increase for the attribute on the treatment set and the estimated probability increase on the control set.

Recent work by Rzepakowski and Jaroszewicz [29] suggests that performance of a tree-based uplift model may improve by using a divergence statistic. The authors propose three postulates that should be obeyed by tree-based splitting criteria. First, the value of the splitting criterion is minimum if and only if the class distributions in treatment and control groups are the same in all branches. Second, splitting criterion is zero if treatment and control are independent. Third, if the control group is empty, the criterion reduces to the case measure. They introduce two new statistics, one based on Kullback-Leibler

divergence, the other based on Euclidean distance. Evaluation on prepared data suggests improved performance. Radcliffe and Surry [28] criticize one of the postulates and the fact that the measures are independent of population size, a parameter that they consider crucial in practical applications.

We observe that the task of discriminating between two dataset strata is closely related to the problem of Relational Subgroup Discovery (RSD), that is, "given a population of individuals with some properties, find subgroups that are statistically interesting" [37]. In the context of multi-relational learning systems, RSD applies a first propositionalization step and then applies a weighted covering algorithm to search for rules that can be considered to define a sub-group in the data. Although the weighting function is defined to focus on unexplored data by decreasing the weight of covered examples, RSD does not explicitly aim at discovering the differences between given partitions.

3 Differential Predictive Concept Definition

Given data that can be partitioned into a set of strata, we define a differential predictive concept as a concept whose measure is significantly different over one stratum as compared to the others. To be more precise, we define a stratified dataset as one composed of disjoint partitions, where each partition contains at least one instance of each target class.

Definition 1 (Stratified Dataset). *Let tc be a target class defined over the set of instances X, and let $D = \{\langle x, tc(x)\rangle\}$ be a set of training examples labeled according to tc. Let $\{D_1, \ldots, D_n\}$ be n disjoint subsets of D, and let D_i^l be the set of training examples of D_i with class label l, such that:*

$$(\forall (i,j) \in [1,n],\ i \neq j)\ D_i \subset D,\ D_i \cap D_j = \emptyset,\ \forall l\ D_i^l \neq \emptyset. \tag{1}$$

A k-strata dataset $\mathcal{D}$ over the set of instances X is the union of k such subsets D_i, with $2 \leq k \leq n$, such that:

$$\mathcal{D} = \{D_i \mid 1 \leq i \leq k\}. \tag{2}$$

After specifying the instance space, we define a differential predictive concept.

Definition 2 (Differential Predictive Concept). *Let c be a concept over the set of instances X, and let $\mathcal{D}$ be a k-strata dataset. Let $S(c|D_i)$ be the classification performance score for c over the subset D_i. A* stratum-j specific differential predictive concept *is a concept c_j such that:*

$$\forall i \neq j,\ S(c_j|D_j) \gg S(c_j|D_i). \tag{3}$$

Score difference ($\gg$) can be evaluated using statistical significance tests or by comparing against a threshold. In this work we will focus on 2-strata 2-class differential problems.

4 Learning Differential Predictive Rules

This work uses Inductive Logic Programming (ILP) [11] to build the first relational differential classifier. The benefit of using ILP in this context is twofold. First, we can use a first-order logic formulation to represent complex relational patterns spanning the patient and mammogram levels. In our motivating application, we can represent data on one mammogram and relate it to prior mammograms for the same patient. Second, we shall take advantage of ILP's ability to learn easily-comprehensible logical rules.

Used for differential prediction, ILP — as a rule-learning technique — has a major advantage: each individual rule can be viewed as a feature describing a subgroup. We can investigate the performance of each rule on a given dataset, identify rules that only apply to particular data subsets, and isolate subgroups covered by a particular rule. Given a stratified dataset, we can examine the performance of rules on the various strata, and select stratum-specific rules that have significantly different performances across strata.

We propose and evaluate three different approaches to learn differential predictive rules. All three approaches can be applied to any ILP algorithm, and can be used with any scoring function S. We use m-estimate to represent the probability of an example given a rule. We set both m and the minimum number of positive examples to be covered by an acceptable clause to 10% of the number of positive examples per stratum and class.

An important concern in real-life situations is population size [28]. Probability estimates tend to favor highly precise estimates (even taking into account the m count) and may be prone to overfitting, a difficult problem in ILP given the number of rules we generate and their complexity. In this work, we heuristically compensate for population size by weighing over the rule positive cover on the case set, as shown below.

4.1 Baseline Approach

As a running example, suppose we are given a 2-strata 2-class dataset of breast cancer records, with class labels *in_situ* and *invasive*, and strata *older* and *younger*. Our task is to find rules that exhibit a differential performance over the two strata. More precisely, we want rules that correctly predict *in_situ* versus *invasive* in the older stratum, but have a significantly worse performance over the younger stratum. Our target stratum D_t is thus *older*, while *younger* is the other stratum D_o.

A simple approach is to merge both strata together while including the stratifying attribute as an additional predicate in the background knowledge. Thus older stratum examples will have $stratum(Example, \texttt{older})$ as an additional feature, while $stratum(Example, \texttt{younger})$ will describe younger instances. We run ILP over the whole dataset and select theory rules that have the condition $stratum(Example, \texttt{older})$ in their body. Such rules are specific to the older stratum. We call this approach the *baseline* approach (BASE).

We score each rule R by considering its positive cover and m-estimate over the merged strata:

$$S_{BASE}(R|D_t, D_o) = poscover(R|D_t \cup D_o) \times mestimate(R|D_t \cup D_o). \quad (4)$$

4.2 Model Filtering Approach

Our second method is a *model filtering* (MF) approach based on [25]. It follows similar principles to the Two Model approach [21,28]. We start by constructing a predictive ILP model over a given stratum. The model outputs a high-performance stratum-specific theory. By construction, the theory rules perform well on their stratum, according to a given scoring function S. We test each theory rule on the other stratum, and select rules with a poor performance, hence filtering the original model. According to this model, the greater the performance difference, the more differential predictive a rule should be.

Fig. 1 flowchart outlines the construction of in situ rules specific to the `older` stratum. Starting with the older subset, we construct an ILP model that discriminates between *in_situ* and *invasive*. The generated rules are expected to have good performance over the `older` stratum. We then test each rule on the `younger` stratum, and keep rules that perform poorly.

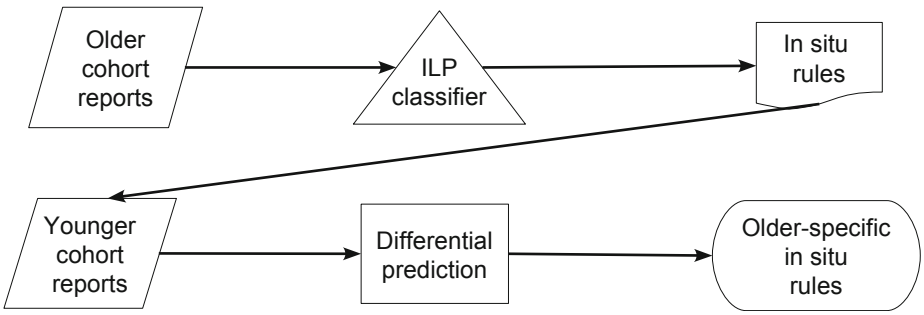


Fig. 1. Model Filtering approach to identify older-specific in situ rules

During the MF search phase, we score a rule R over strata D_t using $S_{BASE}(R|D_t)$. Given the final theory, we score each theory rule R_t according to:

$$S_{MF}(R_t|D_t, D_o) = S_{BASE}(R_t|D_t) - S_{BASE}(R_t|D_o). \quad (5)$$

4.3 Differential Prediction Search Approach

Our third method, *differential prediction search* (DPS), builds a differential prediction ILP classifier by altering the ILP search. Unlike our generate-then-test model filtering method, DPS uses *test-incorporation* by altering the ILP search space. It defines a new clause evaluation function that considers both strata

during search-space exploration and rule construction. This allows ILP to return rules specifically selected for their differential prediction score, that it would have overlooked otherwise. This is achieved through a differential-prediction-sensitive score that measures the performance difference of a rule over both strata.

Definition 3 (Differential-Prediction-Sensitive Scoring). *Let R be a clause (rule) over the set of instances X, and let $\mathcal{D}$ be a 2-strata dataset over X. We define a* differential-prediction-sensitive scoring function *Q as a function of R, D_t and D_o, such that Q is positively correlated to the performance of R over D_t, and negatively correlated to the performance of R over D_o.*

For the DPS method, we introduce the following differential-prediction-sensitive scoring function:

$$Q_{DPS}(R|D_t, D_o) = poscover(R|D_t) \times (mestimate(R|D_t) - mestimate(R|D_o)). \tag{6}$$

Note that this function is non-monotonic, as are most user-defined scoring functions, which prohibits us from custom-pruning the search space.

It is enlightening to relate this scoring function with the postulates described in [29]. Postulate 2 is trivially satisfied: if the condition is independent from treatment than the measure should indeed be zero. In contrast to postulate 1, we select rules that do *better* in one strata, and not rules that do *differently*. This is standard in ILP, where the search aims at covering the positive examples, E^+. In fact, in this setting, the standard techniques to explain negatives is to perform another search, switching E^+ and E^-. The last postulate concerns the case where the control set is empty. In this case, this measure indeed reduces to a classic non-differential ILP scoring function.

Our work thus obeys the main postulates followed by prior work in uplift modeling. Regardless, we observe that, to the best of our knowledge, this the first approach directly designed to learn differential rules. Instead, prior work on differential prediction has focused on learning trees or logistic regression models that can estimate differential performance. Instead, our work focuses on understanding factors that describe differential performance.

Fig. 2 flowchart outlines the construction of older-specific in situ rules. The differential-prediction classifier takes both strata as input. It constructs, scores and selects rules according to their differential-prediction-sensitive score.

5 Experimental Setting

We implement our three differential predictive rule learning methods using Aleph [34]. We invoke *induce_max*, which induces a theory that is unaffected by the order of the examples. We set $depth = 100000$, $i = 10$, $nodes = 50000$ and $clauselength = 5$. We perform experiments with the YAP Prolog compiler [32].

When using synthetic data, we know the ground truth. We then can compare the predicted rules to the original rules. We consider identical rules (up to variable renaming) as true findings. We label the remaining theory rules as

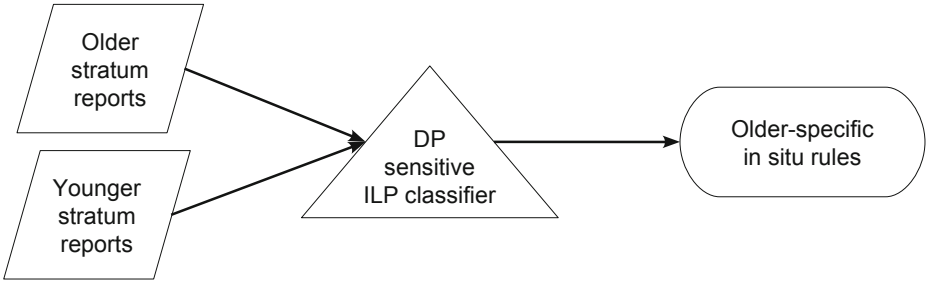


Fig. 2. Differential prediction search approach to identify older-specific in situ rules

false positive findings, and the missing original rules as false negative findings. We rank the theory rules by their score, and compute their precision-recall (PR) curve using [10]. Since we do not have scores associated with the missing false negative findings, we truncate the PR curve at the recall returned by the theory. Note that this yields a PR curve on recovered rules rather than on data.

We compare the different classifiers using their PR area under the curve (AUC-PR). We use the Mann-Whitney test to compare two sets of experiments. When comparing multiple sets, we use the Friedman test with a Hommel adjusted two-tailed Wilcoxon for the post-hoc pairwise tests. We chose these tests based on the recommendation of [12]. We set the confidence level to 95%.

Lacking differential rule ground truth, we can not use this method for real world data. Uplift curves are often used to address this problem [29]. Using 5-folds cross-validation, we use the learned theory rules as attributes to a TAN classifier [14] to assign a probability to each example. Given a threshold p, we compute the lift L_i, defined as the number of positive examples amongst the fraction p of examples that are ranked the highest on strata i. We generate an uplift curve by ranging p from 0 to 1 and plotting $\{p, L_1 - L_2\}$.

6 Synthetic Dataset

Before going to our target application, we use synthetic data to evaluate the ability of our approaches to uncover ground truth differential rules, and to study their sensitivity to variations in noise and in dataset size, two major concerns in real-world data. The multi-relational Michalski-trains dataset [19] is often used by ILP researchers to evaluate system performance in a controlled environment. Given two sets of trains, eastbound and westbound, the original problem consists of finding a concept which explains the eastbound trains. Each train includes multiple carriages of varying size, content and shape. Concept complexity is parametrized by generating more complex explanations of eastbound trains.

To test for differential prediction, we define two categories of trains, *red* and *blue*. We thus have a 2-strata (*red*, *blue*) 2-class (*east*, *west*) dataset. We randomly create up to 5 eastbound rules that are common for both *red* and *blue* trains. We then randomly create two additional sets of eastbound rules, each set

is specific to one stratum, *red* or *blue*. These are color-specific eastbound differential predictive rules. We ensure that all rules are unique, and that color-specific rules are not subsets of common rules nor of each other.

We generate the eastbound trains using the stratum's common and specific rules. We define westbound trains as non-eastbound trains. Our aim is to recover the color *red* differential predictive eastbound rules. They are our target rules.

As an example, suppose we have the following eastbound rules. Common eastbound rule:

$$east(T) \text{ :- } infront(T, C1, C2), short(C1), long(C2). \tag{7}$$

Stratum *red* specific eastbound rule (target rule):

$$east(T) \text{ :- } has\_car(T, C), jagged(C). \tag{8}$$

Stratum *blue* specific eastbound rule:

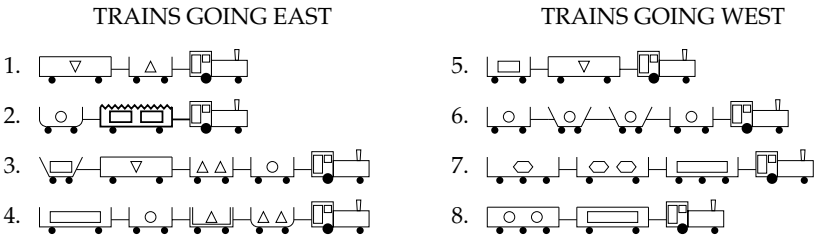
$$east(T) \text{ :- } has\_car(T, C), double(C). \tag{9}$$

Fig. 3(a) shows *red* trains, where eastbound trains 1, 3 and 4 have a short carriage in front of a long one (common rule), while train 2 has a jagged roof carriage (*red* specific rule). Fig. 3(b) shows *blue* trains, where eastbound trains 3 and 4 follow the common rule, while trains 1 and 2 have a double-hulled carriage (*blue* specific rule). Note a jagged roof on *blue* westbound train 5, it would have been classified eastbound if it was *red*.

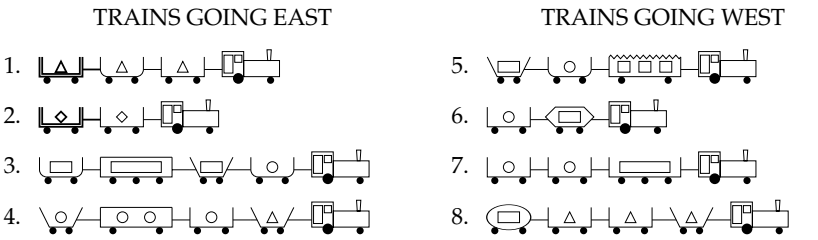
We devise two scenarios, the first with one *red* target rule to recover, and the second with up to 5 *red* target rules. For both scenarios we have up to 5 *blue*-specific rules. For each scenario, we randomly generate 30 different 2-strata 2-class train problems. For every problem, we use a random train generator [24] to randomly construct 1000 eastbound and 1000 westbound trains for each strata, for a total of 4000 trains per experiment. We ensure that each *red* eastbound target rule covers at least 10% of the eastbound *red* trains. We refer to this noise-free data as *clean1000*. To test the scalability of our algorithms, we also construct *clean100*, which consists of the first 100 trains (for each strata, class and problem) of *clean1000*. Since real world data is hardly clean, we also create noisy versions. For each problem, we randomly swap the target class of 5% of our instances, creating the *noisy1000* and *noisy100* datasets.

We end up with 30 simulations for each scenario, noise level, size and method combination. Table 1 reports the AUC-PR mean and standard deviation of each experimental block. When using the *clean* sets, we don't allow any negative examples to be covered by an acceptable clause. When using the *noisy* sets, we allow a negative rule cover of up to 10% of the number of *red* trains.

We compare two methods by using a paired Mann-Whitney test on all their corresponding experiments. Our results show that MF outperforms BASE on all testbeds (p-value = 0.00048). BASE outperforms DPS on size 100 sets (p-value = 0.019), while DPS outperforms BASE on size 1000 (p-value = 0.01). On large noisy sets, DPS outperforms both BASE (p-value = 0.0018) and MF (p-value = 0.0374).



(a) Color *red* trains, specific rule (jagged-roof) in bold



(b) Color *blue* trains, specific-rule (double-hulled) in bold

Fig. 3. A 2-strata 2-class Michalski-train problem

Table 1. AUC-PR mean and standard deviation for each scenario, noise level, size and method combination. Each experimental block is composed of 30 experiments.

Dataset	clean100			clean1000			noisy100			noisy1000		
Method	BASE	MF	DPS	BASE	MF	DPS	BASE	MF	DPS	BASE	MF	DPS
One target rule scenario												
Mean	0.73	**0.83**	0.62	0.87	**0.90**	0.88	0.57	**0.62**	0.54	0.63	0.80	**0.87**
Std dev	0.45	0.34	0.40	0.35	0.24	0.29	0.50	0.47	0.42	0.49	0.36	0.31
Multiple target rules scenario												
Mean	0.61	**0.70**	0.42	0.75	**0.86**	0.77	0.38	**0.52**	0.31	0.52	0.55	**0.65**
Std dev	0.33	0.28	0.29	0.33	0.24	0.30	0.37	0.28	0.32	0.39	0.27	0.29

6.1 Discussion

As one expects, performance improves with larger sets of training examples, and decreases with multiple target rules and noisy sets. The *noisy* runs are harder for three reasons. First is the noise effect *per se*, randomly assigning the wrong target class to 5% of the trains. Second is the 10% minimum positive cover threshold per rule. If a target rule originally narrowly passed this threshold, the addition of noise may decrease its positive coverage below the threshold, and the rule becomes undetectable. Third is the maximum negative cover threshold: in

clean runs, we only consider rules that don't cover any westbound train, which drastically reduces the number of evaluated rules. In *noisy* runs, we allow up to 10% of negative cover. Even if no noise is injected, the exponential expansion of the search space increases the probability that some non-target rule scores better than a target.

It is interesting to note that DPS is the least affected by noise. In each experimental block, DPS suffers the least decrease in mean AUC-PR, none being significant. In the one-target rule and large-set block, adding noise decreases DPS mean by just 1 point, from 0.88 to 0.87 (p-value = 0.94). On the other hand, MF and BASE drop by 10 and 24 percentage points (Table 1). In the four sets of experiments where noise is a variable, DPS drops an average of 8 percentage points, compared to 21.5 for BASE and 20 for MF.

Similarly, DPS improves the most with increasing sample size. In each of the four sets of experiments where size is a variable, DPS displays the highest increase in mean AUC-PR, all of which are significant. In these experiments, DPS increases an average of 32 percentage points, compared to 12 for BASE and 11 for MF (Table 1).

Although no clear pattern emerges from comparing different methods on both one-target and multiple-target scenarios, DPS seems to be slightly more sensitive to the number of target rules. DPS suffers an average decrease of 19 AUC-PR percentage points over the four experimental blocks where target rule scenario is a variable, compared with 13.5 for BASE and 13 for MF (Table 1). Nevertheless, this performance decrease does not alter the method ranking over each experimental block.

In summary, our experiments show that MF is more suitable for either clean data or small datasets. But for large and noisy data, which is what most real world applications are, DPS is more appropriate. In addition, DPS performance increases at a faster rate than MF, and thus may outperform MF for larger clean datasets. DPS, by navigating the differential prediction search space, requires more training examples and generates a set of rules as a consistent theory which explains the data. In contrast, MF and BASE select individual rules that may be suboptimal.

7 Breast Cancer Diagnosis

Our motivating application is to learn older-specific in situ breast cancer differential predictive rules. We apply our three methods to the breast cancer data used in [25]. The data consists of two cohorts: patients younger than 50 years old form the *younger* cohort, while patients aged 65 and above form the *older* cohort. The older cohort has 132 in situ and 401 invasive cases, while the younger one has 110 in situ and 264 invasive.

The data is organized in 20 extensional relations that describe the mammogram, and 35 intensional relations that connect a mammogram with related mammograms, discovered at the same or in prior visits. The background knowledge also maintains information on prior surgeries.

We use the same experimental setting as for the synthetic data, but set $nodes = 200,000$ since the number of predicates is much larger. The BASE method does not return any rule, which highlights the difficulty of this task. Lacking ground truth, we use uplift curves to compare MF and DPS (Fig. 4). DPS consistently outperforms MF, which in turn consistently outperforms a baseline random classifier. DPS has an area under the curve (taken to the baseline) of 16.5, almost double the 9.1 of MF.

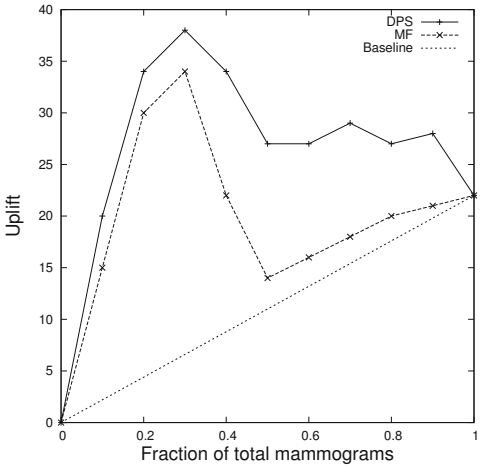


Fig. 4. Uplift curve for breast cancer stage

MF returns 4 differential predictive rules that have a significantly better precision and recall [16] over the older cohort. DPS returns 15. A practicing radiologist, fellowship-trained in breast imaging, examined and assessed all the rules. One MF rule was not found meaningful, while the remaining three are redundant to each other and translate to:

1. Tumor is older-specific in situ if its principal mammographic finding is calcification or single dilated duct, and patient does not have prior surgery.

Single dilated duct is a rare finding and was combined with calcification in our data for convenience. Based on this rule, the more common finding, calcification, is a differential predictor of in situ disease in older patients, which is a novel and interesting result. A possible explanation is that, in asymptomatic women, in situ disease is often associated with screen-detected micro-calcifications; while in symptomatic women, in situ is associated with a palpable mass or pathological nipple discharge [26]. Younger women tend to have more rapidly proliferating cancers that develop into a palpable mass [15], in contrast to more indolent, non-palpable in situ disease manifest as micro-calcification in older patients. This previously unreported finding merits further investigation.

DPS provides a more complete picture of older-specific in situ differential predictors. All 15 returned rules are meaningful and, in addition to extracting the rule described above, four additional themes emerge. DPS is thus able to detect more differentially predictive features than MF, offering a better insight into the medical problem. We select representative clauses from each theme. Tumor is older-specific in situ if:

2. Patient had prior in situ biopsy, and examined-breast had a BI-RADS score of 1 during a previous mammogram, which was not the first visit.
3. Patient had prior in situ biopsy, its examined-breast BI-RADS increased by at least 3 since a previous visit, whereas its other-breast BI-RADS remained constant.
4. Principal mammographic finding is calcification or single dilated duct, examined-breast BI-RADS score increased by at least 3 since a previous visit, and patient had an even earlier screening mammogram.
5. Patient has a breast density of 2, is having a unilateral exam, doesn't have a focal asymmetric density, and principal mammographic finding is calcification or single dilated duct.

Besides calcification, the second DPS rules theme is the presence of a prior in situ biopsy (rules $2, 3$). A prior history of biopsy revealing in situ disease is thus a better predictor of in situ recurrence in older women. This observation is partially explained by the longer life span of older women which offers more time for a recurrence to manifest. But this rule may also relate to the indolent nature of in situ breast cancer in older women. In fact, both invasive and in situ tumors in older patients tend to be less aggressive and have lower rates of local recurrence than tumors in younger patients [15]. More specifically, younger women with in situ disease are more likely to progress to an invasive recurrence rather than develop another in situ tumor when they recur [35].

The third theme is the increase in the examined breast BI-RADS score (rules $3, 4$). The BI-RADS score is a number that summarizes the examining radiologist's opinion and findings concerning the mammogram [4]. The radiologist assigns a score for each examined breast. An increase in the BI-RADS score over multiple visits reflects increasing suspicion of malignancy. This may be a more pronounced feature in older women because they have more prior mammograms.

The next observation, whereas screening visits predict in situ in older women (rule 4), may also relate to the greater opportunity for screening in older patients. Regular screening mammography is usually recommended for women aged 40 and above. Younger women are more likely to seek care for a palpable lump detection rather than via screening [15]. Thus older women tend to have more screening exams because of regular visits after age 40.

Finally we note a class 2 breast density, out of an increasing density scale of 1 to 4 (rule 5). This is a relatively low breast density, more common in older women, since breast density decreases with age [18]. This rule is of special relevance since it doesn't link to any previous mammogram or history predicate, hence leveling the playing field between younger and older in terms of time. It requires a class

2 breast density and an observed calcification during a unilateral (and hence diagnostic) exam. A lower breast density significantly increases mammogram sensitivity [22], allowing for easier micro-calcification detection.

8 Future Work

This work can be extended in several directions. First, our differential prediction search can be tested and validated using a larger experimental set. We can systematically vary the sample size to establish a performance-size curve, and try different scoring functions. We can also fine grain the construction of the Michalski-trains sets by monitoring the coverage of each target or common rule. Noting that we defined westbound as not-eastbound, it would be interesting to gauge model differences if westbound was defined using a separate set of rules.

Second, this work assumes the presence of a stratified dataset. Given a non-stratified dataset, we may be able to select the best dividing attribute that maximizes differential predictive rules performance. We can repeatedly stratify the data using each of its attributes, and perform differential prediction. We then select the stratification achieving the best results. This approach may be used for differential subgroup discovery.

Third, we only proposed solutions for the 2-strata 2-class differential prediction problem. We plan on extending it to multi-strata problems using f-divergence functions. This being the first attempt at relational differential prediction, we can similarly extend our approach to decision-tree learners.

9 Conclusion

In this work, we extend differential prediction to the multi-relational domain using ILP. We devise and implement three methods to learn 2-strata 2-class differential predictive rules. The first baseline method merges the two strata together while including the stratifying attribute as an additional predicate. The model filtering method generates rules on the target stratum and tests them for differential prediction on the other stratum. The differential prediction search approach alters the ILP search space to use a differential-prediction-sensitive scoring function to assess rules over both strata during rule construction. Our experiments over synthetic data show that the model filtering method is more suitable for either clean or small datasets. For large and noisy data, which is what most real world applications are, the differential prediction search method outperforms both the baseline (p-value $= 0.0018$) and the model filtering (p-value $= 0.0374$) approaches. We apply our methods on a breast cancer dataset, and extract novel rules linking calcification to *in situ* disease in older women.

Acknowledgment. This work is supported by US National Institute of Health (NIH) grant R01-CA127379-01. We thank Kendrick Boyd for his help in computing AUC-PR. VSC was funded by the ERDF through the Progr. COMPETE, the Portuguese Gov. through FCT, proj. HORUS ref. PTDC/EIA-EIA/100897/2008,

ADE (PTDC/ EIA-EIA/121686/2010), and the EU Sev. Fram. Progr. FP7/2007-2013 under grant aggrm. 288147.

References

1. Allegra, C.J., Aberle, D.R., Ganschow, P., Hahn, S.M., Lee, C.N., Millon-Underwood, S., Pike, M.C., Reed, S., Saftlas, A.F., Scarvalone, S.A., Schwartz, A.M., Slomski, C., Yothers, G., Zon, R.: National Institutes of Health State-of-the-Science Conference Statement: Diagnosis and Management of Ductal Carcinoma In Situ. J. Natl. Cancer Inst. 102(3), 161–169 (2010)
2. American Cancer Society: Breast Cancer Facts & Figures 2009-2010. American Cancer Society, Atlanta, USA (2009)
3. American Cancer Society: Cancer Facts & Figures 2009. American Cancer Society, Atlanta, USA (2009)
4. American College of Radiology, Reston, VA, USA: Breast Imaging Reporting and Data System (BI-RADS^{TM}), 3rd edn. (1998)
5. American Educational Research Association/American Psychological Association/National Council on Measurement in Education: The Standards for Educational and Psychological Testing (1999)
6. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. Artif. Intell. 101(1-2), 285–297 (1998)
7. Chyou, P.H.: Patterns of bias due to differential misclassification by case control status in a case control study. European Journal of Epidemiology 22, 7–17 (2007)
8. Cleary, T.A.: Test bias: Prediction of grades of negro and white students in integrated colleges. Journal of Educational Measurement 5(2), 115–124 (1968)
9. Davis, J., Burnside, E.S., de Castro Dutra, I., Page, D., Ramakrishnan, R., Santos Costa, V., Shavlik, J.: View Learning for Statistical Relational Learning: With an application to mammography. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, pp. 677–683 (2005)
10. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: Proc. of the 23rd International Conference on Machine Learning, Pittsburgh, PA, pp. 233–240 (2006)
11. De Raedt, L.: Logical and Relational Learning. Springer (2008)
12. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
13. Flegal, K.M., Keyl, P.M., Nieto, F.J.: Differential misclassification arising from nondifferential errors in exposure measurement. American Journal of Epidemiology 134(10), 1233–1244 (1991)
14. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning 29, 131–163 (1997)
15. Gajdos, C., Tartter, P.I., Bleiweiss, I.J., Bodian, C., Brower, S.T.: Stage 0 to stage III breast cancer in young women. J. Am. Coll. Surg. 190(5), 523–529 (2000)
16. Goutte, C., Gaussier, E.: A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In: Proc. of the 27th European Conference on IR Research, pp. 345–359. Santiago de Compostela, Spain (2005)
17. Hansotia, B., Rukstales, B.: Incremental value modeling. Journal of Interactive Marketing 16(3), 35–46 (2002)
18. Kelemen, L.E., Pankratz, V.S., Sellers, T.A., Brandt, K.R., Wang, A., Janney, C., Fredericksen, Z.S., Cerhan, J.R., Vachon, C.M.: Age-specific trends in mammographic density. American Journal of Epidemiology 167(9), 1027–1036 (2008)

19. Larson, J., Michalski, R.S.: Inductive inference of VL decision rules. ACM SIGART Bulletin 63, 38–44 (1977)
20. Linn, R.L.: Single-group validity, differential validity, and differential prediction. Journal of Applied Psychology 63, 507–512 (1978)
21. Lo, V.S.: The true lift model - a novel data mining approach to response modeling in database marketing. SIGKDD Explorations 4(2), 78–86 (2002)
22. Mandelson, M.T., Oestreicher, N., Porter, P.L., White, D., Finder, C.A., Taplin, S.H., White, E.: Breast density as a predictor of mammographic detection: comparison of interval- and screen-detected cancers. J. Natl. Cancer Inst. 92(13), 1081–1087 (2000)
23. Muggleton, S.: Inductive Logic Programming. New Generation Computing 8(4), 295–318 (1991)
24. Muggleton, S.: Random train generator (1998), http://www.doc.ic.ac.uk/textasciitildeshm/Software/GenerateTrains/
25. Nassif, H., Page, D., Ayvaci, M., Shavlik, J., Burnside, E.S.: Uncovering age-specific invasive and DCIS breast cancer rules using Inductive Logic Programming. In: 1st ACM International Health Informatics Symposium, Arlington, VA, pp. 76–82 (2010)
26. Patani, N., Cutuli, B., Mokbel, K.: Current management of DCIS: a review. Breast Cancer Res. Treat 111(1), 1–10 (2008)
27. Radcliffe, N.J., Surry, P.D.: Differential response analysis: Modeling true response by isolating the effect of a single action. In: Credit Scoring and Credit Control VI, Edinburgh, Scotland (1999)
28. Radcliffe, N.J., Surry, P.D.: Real-world uplift modelling with significance-based uplift trees. White Paper TR-2011-1, Stochastic Solutions (2011)
29. Rzepakowski, P., Jaroszewicz, S.: Decision trees for uplift modeling. In: 2010 IEEE International Conference on Data Mining, Sydney, Australia, pp. 441–450 (2010)
30. Sackett, P.R., Laczo, R.M., Lippe, Z.P.: Differential prediction and the use of multiple predictors: The omitted variables problem. Journal of Applied Psychology 88(6), 1046–1056 (2003)
31. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approach to filtering junk e-mail. In: AAAI Workshop on Learning for Text Categorization, Madison, WI (1998)
32. Santos Costa, V.: The life of a logic programming system. In: de la Banda, M.G., Pontelli, E. (eds.) Proceedings of the 24th International Conference on Logic Programming, Udine, Italy, pp. 1–6 (2008)
33. Society for Industrial and Organizational Psychology: Principles for the Validation and Use of Personnel Selection Procedures, 4th edn (2003)
34. Srinivasan, A.: The Aleph Manual, 4th edn. (2007), http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/aleph.html
35. Vicini, F.A., Recht, A.: Age at diagnosis and outcome for women with ductal carcinoma-in-situ of the breast: A critical review of the literature. Journal of Clinical Oncology 20(11), 2736–2744 (2002)
36. Young, J.W.: Differential validity, differential prediction, and college admissions testing: A comprehensive review and analysis. Research Report 2001-6, The College Board, New York (2001)
37. Železný, F., Lavrac, N.: Propositionalization-based relational subgroup discovery with rsd. Machine Learning 62(1-2), 33–63 (2006)

Efficient Training of Graph-Regularized Multitask SVMs

Christian Widmer[1,2], Marius Kloft[3], Nico Görnitz[3], and Gunnar Rätsch[1,2]

[1] Memorial Sloan-Kettering Cancer Center, New York, USA
[2] FML, Max-Planck Society, Tübingen, Germany
[3] Machine Learning Laboratory, TU Berlin, Germany

Abstract. We present an optimization framework for graph-regularized multi-task SVMs based on the *primal* formulation of the problem. Previous approaches employ a so-called multi-task kernel (MTK) and thus are inapplicable when the numbers of training examples n is large (typically $n < 20,000$, even for just a few tasks). In this paper, we present a primal optimization criterion, allowing for general loss functions, and derive its dual representation. Building on the work of Hsieh et al. [1,2], we derive an algorithm for optimizing the large-margin objective and prove its convergence. Our computational experiments show a speedup of up to *three orders of magnitude* over LibSVM and SVMLight for several standard benchmarks as well as challenging data sets from the application domain of computational biology. Combining our optimization methodology with the COFFIN large-scale learning framework [3], we are able to train a multi-task SVM using over 1,000,000 training points stemming from 4 different tasks. An efficient C++ implementation of our algorithm is being made publicly available as a part of the SHOGUN machine learning toolbox [4].

1 Introduction

The main aim of multi-task learning [5] is to leverage the information of multiple, mutually related learning tasks to make more accurate predictions for the individual tasks. For example in computational biology, multiple organisms share a part of their evolutionary history and thus contain related information that can be exploited to mutually increase the quality of predictions (see, e.g., [6,7]). Further examples of successful application domains for multi-task learning include natural language processing [8] (each speaker giving rise to a task) or computer vision [9,10], where multiple visual object classes may share some of the relevant features [11].

Recently, there has been much research revolving around *regularization-based* multi-task learning machines, which, given training points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$, each associated with a task $t(i) \in \{1 \ldots, T\}$, and labels $Y = \{y_1, \ldots, y_n\} \subset \{-1, 1\}$, for each task $t \in \{1, \ldots, T\}$ learn a linear hypothesis $\boldsymbol{x} \mapsto \langle \boldsymbol{w}_t, \boldsymbol{x} \rangle$ by

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 633–647, 2012.

solving the following mathematical optimization problem:

$$\min_{w=(\boldsymbol{w}_1;\ldots;\boldsymbol{w}_T)\in\mathbb{R}^{nT}} \quad \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 + J(\boldsymbol{w}) + C\sum_{i=1}^{n} l\left(y_i \boldsymbol{w}_{t(i)}^{\top}\boldsymbol{x}_i\right), \tag{1}$$

where $l : \mathbb{R} \to \mathbb{R}_{+,0}$ is a convex loss function and $J(\boldsymbol{w}_1, ..., \boldsymbol{w}_M)$ denotes an additional regularization term that promotes similarities of the hypotheses associated to the tasks [5,12,13].

One of the most popular approaches to multi-task learning is by [14], who have introduced a *graph-based* regularization framework; in this setting, each task is represented by a node in a graph and the similarities between the tasks are encoded via an adjacency matrix A, which can be used to promote couplings between tasks in (1) by putting:

$$J(\boldsymbol{w}_1, ..., \boldsymbol{w}_M) = \frac{1}{2}\sum_i \sum_j \left\|\boldsymbol{w}_i - \boldsymbol{w}_j\right\|^2 A_{i,j}\,. \tag{2}$$

Evgeniou, Micchelli, and Pontil [14] show that the dual of this formulation boils down to training a standard support vector machine [15,16] using a so-called *multi-task kernel*

$$K_{\mathrm{MTL}}((\boldsymbol{x}, s), (\tilde{\boldsymbol{x}}, t)) = S_{\mathrm{T}}(s, t) \cdot \langle \boldsymbol{x}, \tilde{\boldsymbol{x}} \rangle\,, \tag{3}$$

where $S_{\mathrm{T}}(s, t)$ is a similarity measure induced by the adjacency matrix A.

In the past, this optimization of this formulation has been addressed by decomposition-based SVM solvers such as SVMLight [17] or LibLinear [2,1] in conjunction with the "kernel" defined in (3). However, this strategy is subject to serious limitations, namely large memory requirements that come from storing the kernel matrix. These limitations allow the efficient use of multi-task learning only for a relative small number of training examples (typically $n < 20,000$, even for a small number of tasks). For larger sample sizes, strategies such as on-the-fly computation of kernel products must be used, which, however, can substantially increase the execution time.

Such large-scale learning problems are frequently encountered nowadays: for example in sequence biology, millions of examples are available from the genomes of multiple organisms and the biological interactions to be learned are typically very *complex*, so that many training examples are needed to obtain a good fit (the lack of sufficient training data is often the main bottleneck in computational biology and multi-task learning). In this paper, we address these limitations by proposing a new optimization framework and giving a high-performance implementation, which is capable of dealing with *millions* of training points at the same time.

In a nutshell, the contributions of this paper can be summarized as follows:

- We present a unifying framework for graph-regularized multi-task learning allowing for arbitrary loss functions and containing, e.g., the works of [12,14] as a special case.

- We give a general dual representation and use the so-obtained primal-dual relations to derive an efficient, provable convergent optimization algorithm for the corresponding large-margin formulation that is based on dual coordinate descent.
- A variety of computational experiments on synthetic data and proven real-world benchmark data sets as well as challenging learning problems from computational genomics show that our algorithms outperform the state-of-the-art by up to three orders of magnitude.
- By including the recent COFFIN framework [3] into our new methodology, we are, for the first time, able to perform graph-based MTL training on very large splice data set consisting of millions examples from 4 organisms.

2 A Novel View of Graph-Regularized Multi-Task Learning

All methods developed in this paper are cast into the established framework of graph-regularized multi-task learning (GB-MTL) outlined in the introduction. Note that Eq (2) may be expressed as

$$\text{Eq. (2)} = \frac{1}{2}\sum_i \sum_j \|w_i - w_j\|^2 A_{i,j} = \sum_i \sum_j w_i^T w_j L_{i,j}\,, \tag{4}$$

where $L = D - A$ denotes the graph Laplacian corresponding to a given similarity matrix A and $D_{i,j} := \delta_{i,j} \sum_k A_{i,k}$. The matrix A is of crucial importance here as it encodes the similarity of the tasks. Note that the number k of zero eigenvalues of the graph Laplacian corresponds to the number of connected components. For the scenario that we are interested in, this will be 1, always.

2.1 Primal Formulation

Using (4), we can thus re-write our base problem (1) as follows:

Generalized *Primal* MTL Problem. *Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^m$ be training data points, each denoted by a task $t(i) \in \{1, \ldots, T\}$, and let $l : \mathbb{R} \to \mathbb{R}$ be a convex loss function. Then the primal MTL optimization problem is given by*

$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_T \in \mathbb{R}^m} \quad \frac{1}{2}\sum_{t=1}^{T} \|\boldsymbol{w}_t\|_2^2 + \frac{1}{2}\sum_{s=1}^{T}\sum_{t=1}^{T} L_{st}\boldsymbol{w}_s^\top \boldsymbol{w}_t + C\sum_{i=1}^{n} l\left(y_i \boldsymbol{w}_{t(i)}^\top \boldsymbol{x}_i\right)\,. \tag{5}$$

A first problem we face is that, when applying the standard Lagrangian formalism and invoking the KKT conditions, there are couplings in between the $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$. Unfortunately, this hinders expressing the $\boldsymbol{w}_t$ solely in terms of the coordinate-wise gradient of the dual objective, which is the core idea behind recently proposed optimization strategies in SVM research that we wish to exploit [1]. As a remedy, in this paper, we propose an alternative approach that is based on the following two improvements:

- First, we deploy a new dualization technique that based on the combination of Lagrangian duality with Fenchel-Legendre conjugate functions, extending the work of [18]. The so-obtained synergy allows us to derive the dual in a cleaner way than it would have been using Lagrangian duality alone.
- Second, we use the "block vector view", which—in combination with the above improvement—allows us to formulate a representer theorem that can be resolved for $\boldsymbol{w}$.

As it turns out, the combination of the above two ingredients allows us to express the weights $\boldsymbol{w}_t$ in terms of the gradients of the dual objective in a very simple way.

2.2 "Block-Vector/Matrix" View

We define $\boldsymbol{w} = (\boldsymbol{w}_1^\top, \ldots, \boldsymbol{w}_T^\top)^\top$ and $\psi : \mathbb{R}^m \mapsto \mathbb{R}^{mT}$ is the canonical injective mapping that maps a data point $\boldsymbol{x}_i \in \mathbb{R}^m$ to a vector in $\mathbb{R}^{mT}$ that is zero everywhere except at the task(i)-th block, i.e., $\psi(\boldsymbol{x}_i)$ looks like as follows:

$$\psi(\boldsymbol{x}_i) := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{x}_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \leftarrow t(i)\text{-}th \ block \tag{6}$$

For example, if $\boldsymbol{x}_i$ belongs to the first task, i.e. $t(i) = 1$, then we have $\psi(\boldsymbol{x}_i) = (\boldsymbol{x}_i, 0, \ldots, 0)^\top$, while, if $\boldsymbol{x}_i$ belongs to the last task, i.e. $t(i) = T$, then $\psi(\boldsymbol{x}_i)$ is is of the form: $\psi(\boldsymbol{x}_i) = (0, \ldots, 0, \boldsymbol{x}_i)^\top$.

Similarly, for a matrix $B \in \mathbb{R}^{T \times T}$, we define

$$\text{block}(B) := \begin{pmatrix} \text{diag}(b_{11}) & \cdots & \text{diag}(b_{1T}) \\ \vdots & & \vdots \\ \text{diag}(b_{T1}) & \cdots & \text{diag}(b_{TT}) \end{pmatrix}, \tag{7}$$

where $\text{diag}(b_{st})$ is a diagonal matrix in $\mathbb{R}^{m \times m}$ with entries b_{st} at the diagonal and zeros everywhere else, i.e., the resulting matrix $\text{block}(B)$ is an element of $\mathbb{R}^{mT \times mT}$.

We can thus very elegantly write our primal problem (5) in terms of the block notation as follows:

Generalized Primal MTL Problem *(Block View).*

$$\min_w \quad \frac{1}{2} \boldsymbol{w}^\top block(I + L)\boldsymbol{w} + C \sum_i l\left(y_i \boldsymbol{w}^\top \psi(\boldsymbol{x}_i)\right), \tag{8}$$

where I is the identity matrix in $\mathbb{R}^{T \times T}$.

Table 1. Loss functions and regularizers used in this paper and corresponding conjugate functions

	loss $l(t)$ / regularizer $g(\boldsymbol{w})$	dual loss $l^*(t)$ / conjugate regularizer $g^*(\boldsymbol{w})$
hinge loss	$\max(0, 1-t)$	t if $-1 \leq t \leq 0$ and ∞ else
ℓ_p-norm	$\frac{1}{2}\|\boldsymbol{w}\|_p^2$	$\frac{1}{2}\|\boldsymbol{w}\|_{p^*}^2$ where $p^* = \frac{p}{p-1}$
quadratic form	$\frac{1}{2}\boldsymbol{w}^\top B\boldsymbol{w}$	$\frac{1}{2}\boldsymbol{w}^\top B^{-1}\boldsymbol{w}$

2.3 Dualization

Now, the above (block-view-) form of the MTL primal allows to derive the Fenchel dual as follows:

$$\begin{aligned}
\text{Eq. (8)} = \quad & \min_{\boldsymbol{w},\boldsymbol{t}} \quad [\frac{1}{2}\boldsymbol{w}^\top \text{block}(I+L)\boldsymbol{w} + C\sum_i l(t_i)] \\
& \text{s.t.} \quad t_i = y_i \boldsymbol{w}^\top \psi(\boldsymbol{x}_i) \\
\overset{\text{Lagrange}}{=} & \max_{\boldsymbol{\alpha}} \quad \min_{\boldsymbol{w},\boldsymbol{t}} \quad [\frac{1}{2}\boldsymbol{w}^\top \text{block}(I+L)\boldsymbol{w} \\
& \qquad + C\sum_i l(t_i) + \sum_i \alpha_i \left(t_i - y_i \boldsymbol{w}^\top \psi(\boldsymbol{x}_i)\right)] \qquad (9) \\
= \quad & \max_{\boldsymbol{\alpha}} \quad [-C\sum_i \max_{t_i} \left(-\frac{\alpha_i t_i}{C} - l(t_i)\right) \\
& - \max_{\boldsymbol{w}} \left(\sum_i \alpha_i y_i \boldsymbol{w}^\top \psi(\boldsymbol{x}_i) - \frac{1}{2}\boldsymbol{w}^\top \text{block}(I+L)\boldsymbol{w}\right)].
\end{aligned}$$

We now make use of the notion of the Fenchel conjugate of a function f, that is $f^*(\boldsymbol{x}) := \sup_{\boldsymbol{y}} \boldsymbol{x}^\top \boldsymbol{y} - f(\boldsymbol{y})$ to derive a general dual form. Note that the Fenchel conjugates of many functions are known from the literature (see Table 1 for conjugates relevant for this paper; cf. [18] for further reading). For example, the conjugate of the function $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_B^2 := \frac{1}{2}\boldsymbol{x}^\top B\boldsymbol{x}$ is $f^*(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_{B^{-1}}^2 = \frac{1}{2}\boldsymbol{x}^\top B^{-1}\boldsymbol{x}$ and the conjugate of the hinge loss $l(t) = \max(0, 1-t)$ is $l^*(t) = t$ if $-1 \leq t \leq 0$ and ∞ else.

We are now ready to proceed with the derivation:

$$\begin{aligned}
\text{Eq. (8)} = \quad & \max_{\boldsymbol{\alpha}} \quad [\underbrace{-\max_{\boldsymbol{w}} \left(\sum_i \alpha_i y_i \boldsymbol{w}^\top \psi(\boldsymbol{x}_i) - \frac{1}{2}\|\boldsymbol{w}\|_{\text{block}(I+L)}^2\right)}_{=\frac{1}{2}\left\|\sum_i \alpha_i y_i \psi(\boldsymbol{x}_i)\right\|^2_{(\text{block}(I+L))^{-1}}} \\
& - C\sum_i \underbrace{\max_{t_i} \left(-\frac{\alpha_i t_i}{C} - l(t_i)\right)}_{=l^*(-\frac{\alpha_i}{C})}] \\
= \quad & \max_{\boldsymbol{\alpha}} \quad [-C\sum_i l^*\Big(-\frac{\alpha_i}{C}\Big) - \frac{1}{2}\left\|\sum_i \alpha_i y_i \psi(\boldsymbol{x}_i)\right\|^2_{\text{block}((I+L)^{-1})}]
\end{aligned}$$

where we used the definition of the Fenchel conjugate and the fact that, clearly, for any matrix B it holds

$$(\text{block}(B))^{-1} = \text{block}(B^{-1})\,.$$

We thus obtain the following MTL dual optimization problem:

General *Dual* MTL Problem. *The dual MTL problem is given by:*

$$\max_{\boldsymbol{\alpha}} \quad -C\sum_i l^*\Big(-\frac{\alpha_i}{C}\Big) - \frac{1}{2}\Big\|\sum_i \alpha_i y_i \psi(\boldsymbol{x}_i)\Big\|^2_{block(M)} \tag{10}$$

where

$$M := (I + L)^{-1} \tag{11}$$

2.4 Special Case: Large-Margin Learning

We can now employ specific loss functions in the primal (5) and obtain a corresponding dual representations right away by plugging the Fenchel conjugate into (10). For example, for the hinge loss, from Table 1 we obtain the conjugate of $l(t) = \max(0, 1-t)$ is $l^*(t) = t$, if $-1 \le t \le 0$ and ∞ else. Clearly, the minimum in (12) will never be attained for the objective being ∞ (take, e.g., $\boldsymbol{w} = \mathbf{0}$ in (5) to obtain a finite upper bound on the optimal objective) so that the left-hand term $\sum_i l^*\big(-\frac{\alpha_i}{C}\big)$ translates into the hard constraints

$$\forall i: \quad 0 \le \alpha_i \le C\,.$$

Moreover, by (7), we have

$$\frac{1}{2}\Big\|\sum_i \alpha_i y_i \psi(\boldsymbol{x}_i)\Big\|^2_{\text{block}(M)} = \frac{1}{2}\sum_{s,t=1}^{T} m_{st}\boldsymbol{w}_s^\top \boldsymbol{w}_t,$$

where $M = (m_{st})_{1\le s,t\le T}$, so that we obtain the following dual problem for the hinge loss:

Dual MTL-SVM Problem. Denote by $M := (I+L)^{-1}$. Then the dual MTL-SVM problem is given by:

$$\max_{\mathbf{0}\le\boldsymbol{\alpha}\le\mathbf{C}} \quad \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}\Big\|\sum_i \alpha_i y_i \psi(\boldsymbol{x}_i)\Big\|^2_{\text{block}(M)} \tag{12}$$

2.5 A Representer Theorem

By the KKT condition *Stationarity*, it follows from (9) that

$$\nabla_{\boldsymbol{w}}\left(\sum_i \alpha_i y_i \boldsymbol{w}^\top\psi(\boldsymbol{x}_i) - \frac{1}{2}\boldsymbol{w}^\top\text{block}(I+L)\boldsymbol{w}\right) = 0\,,$$

which, by (11), translates to

$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{w}^\top M \psi(\boldsymbol{x}_i) \tag{13}$$

and (recalling the definitions (6) and (7)) can be equivalently written as

$$\boldsymbol{w}_t = \sum_{i=1}^{n} m_{t,t(i)} \alpha_i y_i \boldsymbol{x}_i . \tag{14}$$

3 Optimization Algorithms

In order to solve the optimization problem (12), we define:

$$\forall t = 1, \ldots, T: \quad \boldsymbol{v}_t = \sum_{i \in I_t} \alpha_i y_i \boldsymbol{x}_i \,, \tag{15}$$

where $I_t \subset \{1, \ldots, n\}$ denotes the indices of the data points of task t. We thus associate each task t with a "virtual weight vector" $\boldsymbol{v}$ that can be expressed solely terms of the support vectors corresponding to the respective task. Importantly, all the information we need to compute $\boldsymbol{w}$ is contained in $\boldsymbol{v} := (\boldsymbol{v}_1^\top, \ldots, \boldsymbol{v}_T^\top)^\top$, since by (14) holds

$$\forall 1, \ldots, T: \quad \boldsymbol{w}_t = \sum_{s=1}^{T} m_{s,t} \boldsymbol{v}_s . \tag{16}$$

If there is just a single task, as for standard SVM, i.e., $T = 1$ and $M = I$ (because $L = \mathbf{0}$), then the above definition is simply

$$\boldsymbol{w} = \boldsymbol{v} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \,,$$

which is precisely the representation exploited by [1].

3.1 Derivation of the Optimization Algorithm

The basic idea of our *dual coordinate descent* strategy is to optimize one example weight α_i per iteration, project it onto its feasible set and then update the corresponding parameter vector $\boldsymbol{v}_t$ accordingly. In particular, we can perform *dual coordinate descent* as follows: for each $i \in \{1, \ldots, T\}$ we solve

$$
\begin{aligned}
&\operatorname*{argmax}_{d:0\leq\alpha_i+d\leq C} d + \mathbf{1}^\top \boldsymbol{\alpha} \\
&- \frac{1}{2}\sum_{s,t=1}^{T} m_{st}\left(\boldsymbol{v}_s + dy_i\boldsymbol{x}_i\mathbb{1}_{t(i)=s}\right)^\top\left(\boldsymbol{v}_t + dy_i\boldsymbol{x}_i\mathbb{1}_{t(i)=t}\right) \\
&= \operatorname*{argmax}_{d:0\leq\alpha_i+d\leq C} d - \frac{1}{2}\Bigg(m_{t(i),t(i)}\left\|\boldsymbol{v}_{t(i)} + dy_i\boldsymbol{x}_i\right\|^2 \\
&\quad + 2\sum_{s:s\neq t(i)} m_{s,t(i)}\boldsymbol{v}_s^\top\left(\boldsymbol{v}_{t(i)} + dy_i\boldsymbol{x}_i\right)\Bigg) \\
&= \operatorname*{argmax}_{d:0\leq\alpha_i+d\leq C} d - \Bigg(m_{t(i),t(i)}\left(dy_i\boldsymbol{v}_{t(i)}^\top\boldsymbol{x}_i + \frac{1}{2}d^2\boldsymbol{x}_i^\top\boldsymbol{x}_i\right) \\
&\quad + \sum_{s:s\neq t(i)} m_{s,t(i)}y_i\boldsymbol{v}_s^\top\boldsymbol{x}_i d\Bigg) \\
&= \operatorname*{argmax}_{d:0\leq\alpha_i+d\leq C} d - \frac{1}{2}d^2\boldsymbol{x}_i^\top\boldsymbol{x}_i - \sum_{s=1}^{T} m_{s,t(i)}y_i\boldsymbol{v}_s^\top\boldsymbol{x}_i d
\end{aligned}
$$

We thus observe that for the gradient it holds

$$
\frac{\partial f(\boldsymbol{\alpha} + d\boldsymbol{e}_i)}{\partial d} = 1 - d\boldsymbol{x}_i^\top\boldsymbol{x}_i - \sum_{s=1}^{T} m_{s,t(i)}y_i\boldsymbol{v}_s^\top\boldsymbol{x}_i = 0
$$

which is equivalent to

$$
d = \frac{1 - \sum_{s=1}^{T} m_{s,t(i)}y_i\boldsymbol{v}_s^\top\boldsymbol{x}_i}{\boldsymbol{x}_i^\top\boldsymbol{x}_i}\,. \tag{17}
$$

Therefore, taking the needed projections onto the constraints into account, we have the following update rule in each coordinate descent step:

$$
\alpha_i = \max\left(0, \min\left(C, \alpha_i + d\right)\right). \tag{18}
$$

Note that, if there is only a single task, then $L = 0$ and thus $M = I$, where I is the identity matrix, and we hence obtain the usual LibLinear standard update (denoting $\boldsymbol{w} = \boldsymbol{v} = \boldsymbol{v}_1$):

$$
d = \frac{1 - y_i\boldsymbol{w}^\top\boldsymbol{x}_i}{\boldsymbol{x}_i^\top\boldsymbol{x}_i}\,.
$$

The resulting training algorithm is shown in Algorithm (1).

3.2 Convergence Analysis

To prove convergence of our algorithms, we phrase the following useful result about convergence of the (block-) coordinate descent method:

Algorithm 1. (MULTI-TASK LIBLINEAR TRAINING ALGORITHM). Generalization of the LibLinear training algorithm to multiple tasks.

input: $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^m$, $t(1), \ldots, t(n) \in \{1, \ldots, T\}$, $y_1, \ldots, y_n \in \{-1, 1\}$
for all $i \in \{1, \ldots, n\}$ initialize $\alpha_i = 0$
for all $t \in \{1, \ldots, T\}$ put $\boldsymbol{v}_t = \sum_{i \in I_t} \alpha_i y_i \boldsymbol{x}_i$
while optimality conditions are not satisfied **do**
    for all $i \in \{1, \ldots, n\}$
        compute d according to (17)
        store $\hat{\alpha}_i := \alpha_i$
        put $\alpha_i := \max(0, \min(C, \hat{\alpha}_i + d))$
        update $v_{t(i)} := v_{t(i)} + (\alpha_i - \hat{\alpha}_i) y_i \boldsymbol{x}_i$
    end for
end while
for all $t \in \{1, \ldots, T\}$ compute $\boldsymbol{w}_t$ from $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_T$ according to (16)
output: $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_T$

Proposition 1 (Bertsekas, 1999, Prop. 2.7.1). *Let $\mathcal{X} = \bigotimes_{m=1}^{M} \mathcal{X}_m$ be the Cartesian product of closed convex sets $\mathcal{X}_m \subset \mathbb{R}^{d_m}$, be $f : \mathcal{X} \to \mathbb{R}$ a continuously differentiable function. Define the nonlinear block Gauss-Seidel method recursively by letting $\boldsymbol{x}^0 \in \mathcal{X}$ be any feasible point, and be*

$$\boldsymbol{x}_m^{k+1} = \underset{\boldsymbol{\xi} \in \mathcal{X}_m}{\operatorname{argmin}} f\left(\boldsymbol{x}_1^{k+1}, \cdots, \boldsymbol{x}_{m-1}^{k+1}, \boldsymbol{\xi}, \boldsymbol{x}_{m+1}^{k}, \cdots, \boldsymbol{x}_M^{k}\right), \tag{19}$$

for all $m = 1, \ldots, M$. Suppose that for each m and $\boldsymbol{x} \in \mathcal{X}$, the minimum

$$\min_{\boldsymbol{\xi} \in \mathcal{X}_m} f\left(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{m-1}, \boldsymbol{\xi}, \boldsymbol{x}_{m+1}, \cdots, \boldsymbol{x}_M\right) \tag{20}$$

is uniquely attained. Then every limit point of the sequence $\{\boldsymbol{x}^k\}_{k \in \mathbb{N}}$ is a stationary point.

The proof can be found in [19], p. 268-269. We can conclude the following corollary, which establishes convergence of the proposed MTL training algorithm.

Theorem 1. *Let l be the hinge loss. Then every limit point of Algorithm 1 is a globally optimal point of (12).*

Proof. First, note that the objective function in (12) is continuously differentiable and convex. Second, we can without loss of generality replace the constraints $0 \leq \alpha_i$ by $0 \leq \alpha_i \leq \alpha_i^*$ for all i, where $\boldsymbol{\alpha}^*$ denotes the optimal solution of (12). Thus, in order to show that the constraints form a closed set, it suffices to show that $\alpha_i^* < \infty$ for all i. To this end, we note that setting $\boldsymbol{w} = \boldsymbol{0}$, which is a feasible point in the primal (5), lets us conclude that the optimal primal objective is less than or equal to $o := C \sum_{i=1}^{n} l(0) = C \sum_{i=1}^{n} \max(0, 1 - 0) = Cn < \infty$. Hence, denoting by $\boldsymbol{w}^*$ the primal-optimal point, we obtain $\frac{1}{2} \|\boldsymbol{w}^*\| \leq o$ and thus, by (14), it holds $\frac{1}{2} \left\| \sum_i \alpha_i^* y_i \psi(\boldsymbol{x}_i) \right\|^2_{\text{block}(M)} \leq o$, so that we can conclude that the dual objective in (12) in smaller than or equal to $2o < \infty$. From the latter, we can conclude $\alpha *_i \leq 2o < \infty$ for all i, which was sufficient to show.

4 Computational Experiments

In this section, we evaluate the runtime of our proposed dual coordinate descent (DCD) algorithm (described in Algorithm Table 1), which we have implemented[1] (along with a LibLinear-style shrinking strategy) in C++ as a part of the SHOGUN machine learning toolbox [4]. We compare our solver with the state-of-the-art, that is, SVMLight (as integrated into the SHOGUN toolbox) using the multi-task kernel (MTK) as defined in (3).[2]

We experiment on the following five data sets, whose data statistics are summarized in Table 2:

- *Gauss2D.* A controlled, synthetic data set consisting of a balanced sample from two isotropic Gaussian distributions.
- *Breast Cancer.* A classic benchmark data set consisting of a genetic signature of 60 genes used to predict the response to chemotherapy.
- *MNIST-MTL.* A multi-task data set derived from the well-known MNIST data[3] by considering the three separate tasks "1 vs. 0", "7 vs. 9", and "2 vs. 8".
- *Landmine.* A classic multi-task data set, where the different tasks correspond to detecting land mines under various conditions [20].
- *Splicing.* This is the most challenging data set: a huge-scale, multiple-genomes, biological data set, where the goal is to detect splice sites in various organisms, each organism corresponding to a task. The features are derived from raw DNA strings by means of a weighted-degree string kernel [21].

The above data sets are taken from various application domains including computer vision, biomedicine, and computational genomics, and cover many different settings such as small and large dimensionality, various numbers of examples and tasks. Our corpus includes controlled synthetic data as well as established real-world benchmark data and challenging multiple-genomes splice data. The first four data sets contain real valued data, for which we used linear kernels and corresponding standard scalar products.

To compare our implementation with SVMLight using the multi-task kernel (MTK), we measure the *function difference*

$$\Delta := \left| \mathrm{obj}^* - \widehat{\mathrm{obj}} \right| ,$$

where obj^* the true optimal objective and $\widehat{\mathrm{obj}}$ the actual objective achieved by the solver (for DCD and MTK these are primal and dual objectives, respectively). The true objective obj^* is computed up to a duality gap of $< 10^{-10}$. All experiments are performed on a 4GB AMD64 machine using a single core.

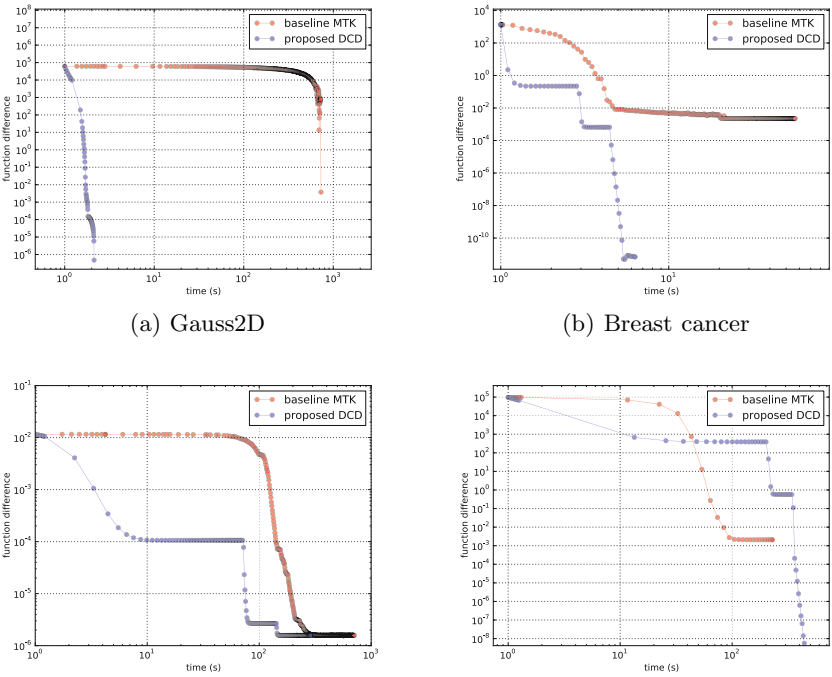
[1] For implementation details, see: `http://bioweb.me/mtl-dcd-solver`

[2] We expect very similar run times by using LIBSVM instead of SVMLight. The runtime measurement was easier to implement in SVMLight than in LIBSVM, which is why we chose the former in our experiments. The SVMLight timing code is specific to our experiments and is therefore located in the ecml2012 git branch of SHOGUN, which is available at: `http://bioweb.me/mtl-dcd`

[3] `http://yann.lecun.com/exdb/mnist/`

Table 2. Statistics of the data sets used in this paper

	dim	#examples	#tasks
Gauss2D	2	$1 \cdot 10^5$	2
Breast Cancer	44	474	3
MNIST-MTL	784	$9.0 \cdot 10^3$	3
Land Mine	9	$1.5 \cdot 10^4$	29
Splicing	$6 \cdot 10^6$	$6.4 \cdot 10^6$	4

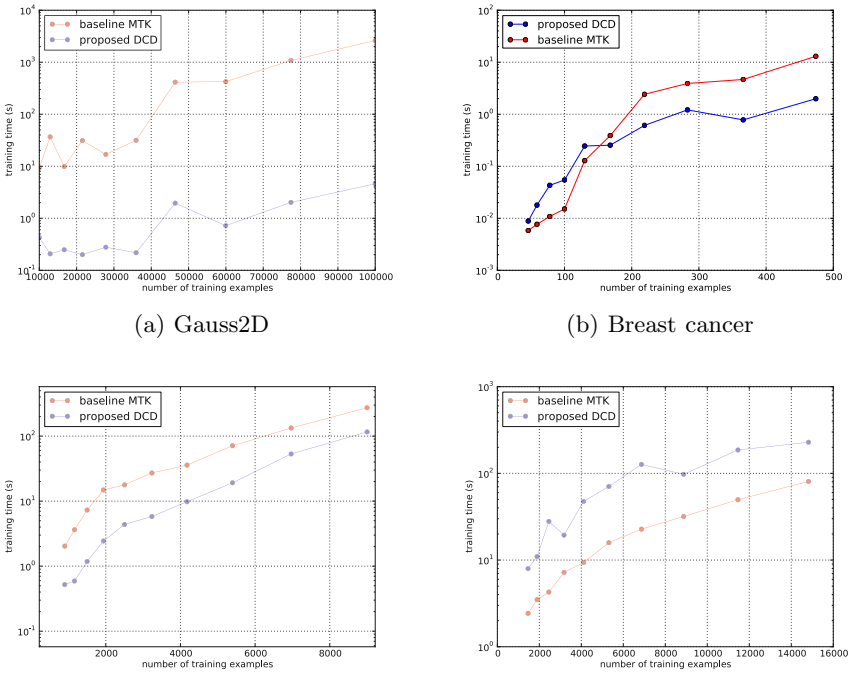


(a) Gauss2D (b) Breast cancer

(c) MNIST-MTL (d) Land Mine

Fig. 1. Results of the runtime experiment in terms of the function difference as a function of the execution time

The results are shown in Figure 1, where the function difference of the four real-valued data sets is shown as a function of the execution time. First of all, we observe that in all four cases the two solvers suffer from an initialization phase, in which the function value improves only slowly. For *Gauss2D* the convergence properties of the two methods (e.g., steepness of the decrease in function difference) are very similar, but our proposed DCD solver being up to three magnitudes faster. Furthermore, we observe that, for two out the four data sets, the MTK baseline fails to decrease the function difference beyond a threshold ranging from 10^{-2} to 10^{-4}, while the proposed DCD algorithm nicely converges to



(a) Gauss2D (b) Breast cancer

(c) MNIST-MTL (d) Land Mine

Fig. 2. Results of the second runtime experiment: required time to train a multi-task SVM to a relative precision of 10^{-4} for various sample sizes n

a precision of 10^{-7} to 10^{-10} (cf. Figure 1(b) – (d)). Finally, we can observe that if we stop both algorithms at some arbitrary time point, our method tends to output a solution that is more precise than the MTK baseline by usually several orders of magnitudes (up to ten orders for, e.g., *Gauss2D, and Breast Cancer*).

In a second experiment, we measure the training time a solver needs to reach a given precision (we chose 10^{-4}) as a function of the training set size. The results of this experiment are shown in Figure 2. We observe that for 3 out of 4 data sets, the proposed DCD methods requires less computation time than the MTK solver. For the synthetic data set the difference is the most drastic, being of the order of up to 2.5 magnitudes. Our method is outperformed by the MTK algorithm on the landmine data set (see Subfigure 2(d)), which indicates that our strategy is in disadvantage if the number of tasks is large relative the number of training examples, due to the update rule given by Equation 17. We expect the curves to cross if there are more training examples per task.

Finally, we study a very large splice data set, where the goal is to detect splice sites in various organisms, each organism corresponding to one task. For the MTK solver, the features are derived from raw DNA strings by means of a weighted-degree string kernel [21] of degree 8; for the DCD solver, we combine

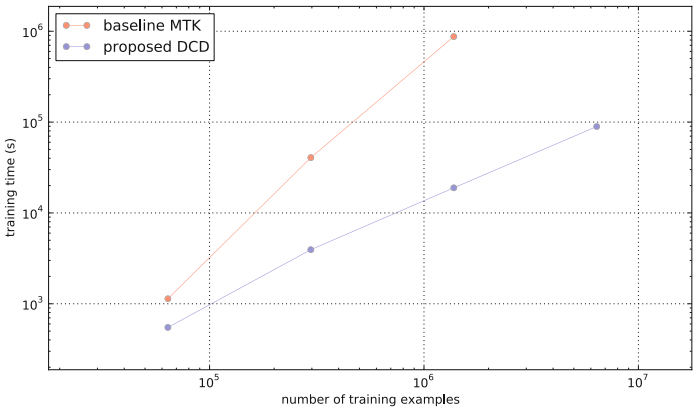


Fig. 3. Results of the large-scale splice site detection experiment

the proposed algorithmic methodology with the COFFIN framework [3] (efficient feature hashing for high-dimensional but sparse feature spaces) as implemented in SHOGUN [4].

The results of this experiment are shown in Figure 3. We observe that the proposed DCD solver is capable of dealing with millions of training points, while the MTK baseline is limited to rather moderate training set sizes of up to hundreds of thousands training points. This experiment demonstrates that we are now able to train on very large genomic sequences in reasonable time, finally allowing for truly large-scale multi-task learning.

5 Conclusion

We have introduced a dual coordinate descent method for graph-regularized multi-task learning. Unlike previous approaches, our optimization methodology is based on the *primal* formulation of the problem. Viewing the latter in terms of block vectors and subsequently deploying Fenchel-Legendre conjugate functions, we derived a general dual criterion allowing us to plug in arbitrary convex loss functions. We presented an efficient optimization algorithm based on dual coordinate descent and prove its convergence. Empirically, we show that our method outperforms existing optimization approaches by up to three orders of magnitude.

By including the recently developed COFFIN framework [3]—which devises feature hashing techniques for extremely high-dimensional feature spaces—into our methodology, we are able, to train a multi-task support vector machine on a splice data set consisting of over $1,000,000$ training examples and 4 tasks. An efficient C++ implementation of our algorithm is being made publicly available as a part of the SHOGUN machine learning toolbox [4].

Our new implementation opens the door to various new applications of multi-task learning in sequence biology and beyond, as it now becomes feasible to combine very large data sets from *multiple* organisms [22]. Our methodology may also serve as technological blueprint for developing further large-scale learning techniques in general: the block vector view gives insights into *structured* learning problems beyond the ones studied in the present paper and, combined with our novel dualization technique, we are able to also extend our optimization approach to various other structured learning machines such as, e.g., structured output prediction as proposed by [7] and block ℓ_p-norm regularized risk minimizers (e.g., [23]).

Acknowledgements. We would like to thank Alexander Zien, who contributed to the *cancer* data set and Jose Leiva for helpful discussions. This work was supported by the German National Science Foundation (DFG) under MU 987/6-1 and RA 1894/1-1 as well as by the European Communitys 7th Framework Programme under the PASCAL2 Network of Excellence (ICT-216886).

References

1. Hsieh, C., Chang, K., Lin, C., Keerthi, S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the 25th International Conference on Machine Learning, pp. 408–415 (2008)
2. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research 9, 1871–1874 (2008)
3. Sonnenburg, S., Franc, V.: Coffin: A computational framework for linear SVMs. In: Fürnkranz, J., Joachims, T. (eds.) ICML, pp. 999–1006. Omnipress (2010)
4. Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., de Bona, F., Binder, A., Gehl, C., Franc, V.: The SHOGUN Machine Learning Toolbox. Journal of Machine Learning Research 11, 1799–1802 (2010)
5. Pan, S., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 1345–1359 (2009)
6. Schweikert, G., Widmer, C., Schölkopf, B., Rätsch, G.: An Empirical Analysis of Domain Adaptation Algorithms for Genomic Sequence Analysis. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems 21, pp. 1433–1440 (2008)
7. Görnitz, N., Widmer, C., Zeller, G., Kahles, A., Sonnenburg, S., Rätsch, G.: Hierarchical Multitask Structured Output Learning for Large-scale Sequence Segmentation. In: Advances in Neural Information Processing Systems 24 (2011)
8. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Cohen, W.W., McCallum, A., Roweis, S.T. (eds.) ICML. ACM International Conference Proceeding Series, vol. 307, pp. 160–167. ACM (2008)
9. Jiang, Y.G., Wang, J., Chang, S.F., Ngo, C.W.: Domain adaptive semantic diffusion for large scale context-based video annotation. In: ICCV, pp. 1420–1427. IEEE (2009)
10. Samek, W., Binder, A., Kawanabe, M.: Multi-task Learning via Non-sparse Multiple Kernel Learning. In: Real, P., Diaz-Pernil, D., Molina-Abril, H., Berciano, A., Kropatsch, W. (eds.) CAIP 2011, Part I. LNCS, vol. 6854, pp. 335–342. Springer, Heidelberg (2011)

11. Torralba, A., Murphy, K.P., Freeman, W.T.: Sharing visual features for multiclass and multiview object detection. IEEE Trans. Pattern Anal. Mach. Intell. 29, 854–869 (2007)
12. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: International Conference on Knowledge Discovery and Data Mining, pp. 109–117 (2004)
13. Agarwal, A., Daumé III, H., Gerber, S.: Learning Multiple Tasks using Manifold Regularization. In: Advances in Neural Information Processing Systems 23 (2010)
14. Evgeniou, T., Micchelli, C., Pontil, M.: Learning multiple tasks with kernel methods. Journal of Machine Learning Research 6(1), 615–637 (2005)
15. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning 20, 273–297 (1995)
16. Müller, K.R., Mika, S., Rätsch, G., Tsuda, K., Schölkopf, B.: An introduction to kernel-based learning algorithms. IEEE Neural Networks 12(2), 181–201 (2001)
17. Joachims, T.: Making large–scale SVM learning practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods — Support Vector Learning, pp. 169–184. MIT Press, Cambridge (1999)
18. Rifkin, R.M., Lippert, R.A.: Value regularization and Fenchel duality. J. Mach. Learn. Res. 8, 441–479 (2007)
19. Bertsekas, D.: Nonlinear Programming, 2nd edn. Athena Scientific, Belmont (1999)
20. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with dirichlet process priors. J. Mach. Learn. Res. 8, 35–63 (2007)
21. Sonnenburg, S., Rätsch, G., Rieck, K.: Large scale learning with string kernels. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) Large Scale Kernel Machines, pp. 73–103. MIT Press, Cambridge (2007)
22. Consortium, T.W.T.C.C.: Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. Nature 447(7145), 661–678 (2007)
23. Kloft, M., Brefeld, U., Sonnenburg, S., Zien, A.: Lp-norm multiple kernel learning. Journal of Machine Learning Research 12, 953–997 (2011)

Geometry Preserving Multi-task Metric Learning

Peipei Yang, Kaizhu Huang, and Cheng-Lin Liu

National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences,
Beijing, China 100190
{ppyang,kzhuang,liucl}@nlpr.ia.ac.cn

Abstract. Multi-task learning has been widely studied in machine learning due to its capability to improve the performance of multiple related learning problems. However, few researchers have applied it on the important metric learning problem. In this paper, we propose to couple multiple related metric learning tasks with von Neumann divergence. On one hand, the novel regularized approach extends previous methods from the vector regularization to a general matrix regularization framework; on the other hand and more importantly, by exploiting von Neumann divergence as the regularizer, the new multi-task metric learning has the capability to well preserve the data geometry. This leads to more appropriate propagation of side-information among tasks and provides potential for further improving the performance. We propose the concept of *geometry preserving probability (PG)* and show that our framework leads to a larger PG in theory. In addition, our formulation proves to be jointly convex and the global optimal solution can be guaranteed. A series of experiments across very different disciplines verify that our proposed algorithm can consistently outperform the current methods.

Keywords: multi-task learning, metric learning, geometry preserving.

1 Introduction

Metric learning has been widely studied in machine learning due to its importance in many machine learning tasks [6,10]. The objective of metric learning is to learn a proper metric function from data, usually a Mahalanobis distance defined as $d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^\top A(\mathbf{x}-\mathbf{y})}$, while satisfying certain extra constraints called side-information, e.g., similar (dissimilar) points should stay closer (further). On the other hand, multi-task learning (MTL), which refers to the joint training of multiple problems, has recently received considerable attention [4,8,11]. If the different problems are closely related, MTL could lead to better performance by propagating information among tasks.

Despite their importance, there are few researches combining multi-task learning with metric learning. To our best knowledge, only recently [8], [12], and [11] developed a multi-task metric learning framework separately. [8] proposed a novel multi-task framework called mtLMNN which directly extends the famous metric learning method Large Margin Nearest Neighbor (LMNN) [10]. Assuming the

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 648–664, 2012.

metric of each task to be a combination of a common and a task-specific metric, mtLMNN proposed to learn the metrics jointly for all the tasks. Exploiting further the Frobenius norm as the regularization term to encourage the similarity among all tasks, mtLMNN indeed showed promising performance in several real datasets. On the other hand, [12] first concatenated all columns of each Mahalanobis matrix A_t for each task t to form a vector $\tilde{A}_t = \mathrm{vec}\,(A_t)$. Tasks are then coupled with each other by $\mathrm{tr}\left(\tilde{A}\Omega^{-1}\tilde{A}^\top\right)$ where $\tilde{A} = [\mathrm{vec}\,(A_1), \ldots, \mathrm{vec}\,(A_T)]$. The author explained this method from a probabilistic viewpoint while failing to validate it empirically. In another aspect, [11] assumed that the useful information of all tasks share a common low-rank subspace. By jointly learning the metrics in this common subspace, the performances of all tasks are improved.

All the above methods have some limitations. When describing the task relationship, the former two methods exploited merely simple vector-based divergence measures. More specifically, if we concatenated all columns of each matrix as a vector, in [8], Frobenius norm between two matrices simply presents the Euclidean distance, while, in [12], the divergence is given as the weighted Euclidean distance. Vector-based divergence may not be powerful enough to measure the relationship between matrices or distance metrics. It cannot preserve the data geometry and will lead to inaccurate information propagation among tasks. For [11], since the formulation is not convex, the global optimal solution is not guaranteed. Besides, the assumption is too strict in some cases.

For a better illustration of the above mentioned phenomenon, we show in Fig. 1 three graphs associated with different distance metrics, determined by a Mahalanobis matrix B, A_1, and A_2 respectively for each graph (from left to right). To visualize the Mahalanobis metric in the Euclidean space, we transform each point $\mathbf{x}_i$ to $\hat{\mathbf{x}}_i = A^{1/2}\mathbf{x}_i$ when plotting so that the Euclidean distance of any pair of transformed points $\|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^2$ is exactly the Mahalanobis distance of the original points $d_A(\mathbf{x}_i, \mathbf{x}_j)$. Geometrically observed, the metric A_2 is obviously more similar to B than to A_1. However, when calculating the similarity using the squared Frobenius norm of difference, surprisingly, A_1 is more similar to B than to A_2! This shows that minimizing Frobenius norm cannot preserve the geometry and hence it may not be appropriate for measuring the divergence of metrics.

Distinct with the above methods, in this paper, we engage the *Bregman matrix divergence* [3] and design a more general regularized framework for multi-task

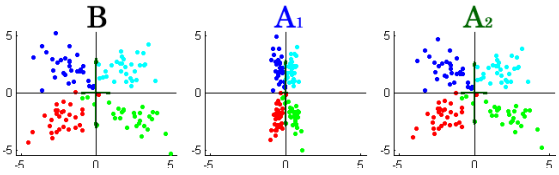


Fig. 1. Illustration of Frobenius norm for metric measurement. Using Frobenius norm, B is more similar to A_1 than to A_2, showing that Frobenius norm cannot preserve the geometry.

metric learning. On one hand, the general framework exploited a more general matrix divergence. We show that it naturally incorporates mtLMNN (using the Frobenius norm) as a special case. On the other hand and more importantly, by exploiting a special Bregman divergence called *von Neumann divergence* [3] as the regularizer, the new multi-task metric learning has the capability to well preserve the geometry when transferring information from one metric to another. We define the *geometry preserving probability* and provide theoretical analysis showing that our new multi-task metric learning method leads to a larger geometry preserving probability and has the capability to better preserve geometry. This enables more appropriate information propagation among tasks and hence provides potentials for further raising the performance. In addition to the geometry preserving property, the new multi-task framework with the von Neumann divergence remains convex, provided that any convex metric learning is used. The novel regularized multi-task metric learning framework is then justified in the probabilistic view point with a series of theoretical analysis. Extensive experimental results across very different disciplines also verify that our proposed algorithm can consistently outperform the current methods.

The rest of this paper is organized as follows. In Section 2, we will present the novel multi-task metric learning framework with Bregman matrix divergence. In Section 3, we present theoretical analysis to show our method can indeed preserve the geometry. In Section 4, we evaluate our method across five real data sets. Finally, we give concluding remarks in Section 5.

2 Novel Regularized Multi-task Metric Learning

In this section, we first present the problem definition and describe the objective of multi-task metric learning formally. Then the concept of *geometry preserving probability* is proposed to give a mathematical measure of the capability to preserve the relative distance between two metrics. After that, we introduce the main work that exploits von Neumann divergence to regularize the relationship among multiple tasks. Finally, we present a practical algorithm to solve the involved optimization problem.

2.1 Problem Definition

A *metric* defined on set $\mathbb{X}$ is a *function* $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+ \doteq [0, +\infty)$ satisfying certain conditions [2]. Denoting the set containing all metrics by $\mathcal{F}_{\mathbb{X}}$ and given any pair of metrics $d_A(\cdot,\cdot), d_B(\cdot,\cdot) \in \mathcal{F}_{\mathbb{X}}$, a divergence function $D : \mathcal{F}_{\mathbb{X}} \times \mathcal{F}_{\mathbb{X}} \to \mathbb{R}_+$ is defined to measure the dissimilarity of d_A and d_B. Since the Mahalanobis metric $d_A(\cdot,\cdot)$ is ultimately determined by the Mahalanobis matrix A, we denote $D(d_A, d_B) \triangleq D(A, B)$ for short.

Assume that there are T related metric learning tasks. For each task-t, its training data set $\mathcal{S}_t$ contains N_t m-dimensional data points $\mathbf{x}_{tk} \in \mathbb{R}^m$ and a triplet set $\mathcal{T}_t = \{(i, j, k) | d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k)\}$. These triplets provide side-information like relative constraints such that $\mathbf{x}_i$ is more similar to $\mathbf{x}_j$ than

to $\mathbf{x}_k$ under the new metric.[1] The objective of multi-task metric learning is to learn T proper Mahalanobis matrices A_t, $t = 1, 2, \ldots T$ jointly and simultaneously. This is significantly different from single-task metric learning where the Mahalanobis matrix is learned independently and isolatedly.

The advantages of learning multiple metrics jointly can be illustrated in Fig. 2 where the famous single task metric learning method LMNN [10] is adopted. Assume different colors indicate the labels of samples and the points with (without) a black border represent training (testing) samples. LMNN attempts to learn a new metric to encourage the neighborhood around every point to stay "pure". For each point, some points with the same label are selected as *targets* ($\triangle$) and any point with different label is expected to stand further than each target with a large margin (the dashed perimeter). Points with different label and lying within the margin are called *imposers* ($\square$). The objective of LMNN is to pull the target nearer and push all imposers outside the margin. Fig. 2(b)/2(f) show the learned metric of task-1/2 where the red/green imposers are pushed away. Unfortunately, when the training samples of green/red class are too few to represent the distribution, some testing samples invade the perimeter in the learned metric of task-1/2. However, as shown in Fig. 2(a) and 2(e), the samples in both tasks have a similar distribution to each other and we expect to improve the performance of both two tasks with help of each other. Appropriate joint metric learning of task-1 and task-2 can lead to an ideal metric for each task. For example, in Fig. 2(c)/2(g), the metric of task-1/2 can be well learned based on our novel geometry preserving framework by pushing away green/red classes with the help of task-2/1 samples. On the other hand, inappropriate multi-task metric learning may not lead to good performance. See Fig. 2(d)/2(h) for example, where the side-information is propagated by squared Frobenius norm of difference of Mahalanobis matrices as mtLMNN did.

2.2 Geometry Preserving between Metrics

In Section 2.1, we have illustrated that jointly learning multiple related metrics could benefit from the geometry preserved from other metrics. In the following, we will propose the mathematical description of the concept of *geometry preserving*.

Since the purpose of metric learning is to refine the distances among different points based on the side-information, when we mention propagating information among tasks by jointly learning multiple metric learning tasks, the information propagated is nothing but the side-information embedded in the metric. On the other hand, in most situations, the side-information specifies the relative distances between different pairs of points rather than their exact distances. For example, one popular kind of side-information is to make similar pairs nearer than dissimilar pairs. Thus, it is more important to propagate the relative distance of points from one task to another. Specifically, assume that we have two

[1] Other settings, e.g., the constraints given by similar and dissimilar pairs could be also used.

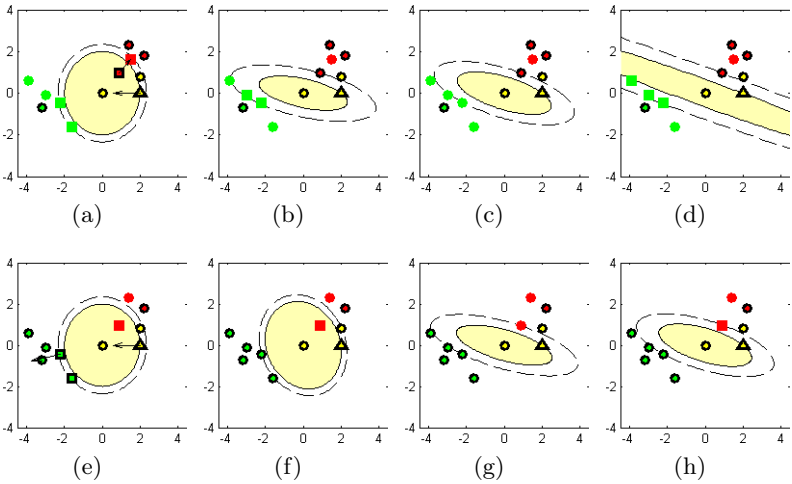


Fig. 2. An illustration of multi-task metric learning. (a/e) The original data of task 1/2. (b/f) The data of task 1/2 after single task metric learning. (c/g) The data of task 1/2 after joint metric learning using von Neumann divergence as regularizer. (d/h) The data of task 1/2 after joint metric learning using squared Frobenius norm of difference as regularizer. Joint learning of multiple tasks (given by our proposed geometry preserving framework) can lead to ideal metrics for both task-1 in (c) & task-2 in (g).

metric learning tasks to learn Mahalanobis matrices A and B respectively. Given $d_B(\mathbf{x}_1, \mathbf{x}_2) < d_B(\mathbf{x}_3, \mathbf{x}_4)$, if we are going to propagate this side-information embedded in d_B to d_A, it is desirable of d_A to make the similar judgement on the relative distance of these two pairs of points, i.e. $d_A(\mathbf{x}_1, \mathbf{x}_2) < d_A(\mathbf{x}_3, \mathbf{x}_4)$. In contrast, the exact absolute values of these distances are less important.

Based on the idea, we propose the concept of *geometry preserving probability* to measure the probability of that the relative distance of arbitrary two pairs of points can be preserved or be consistent for the two metrics.

Definition 1 (Geometry Preserving Probability). *Suppose* $\mathbf{x}_1, \mathbf{y}_1 \in \mathbb{X}$ *and* $\mathbf{x}_2, \mathbf{y}_2 \in \mathbb{X}$ *are two pairs of random points following certain distribution defined by probability density* $f(\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, \mathbf{y}_2)$*. If two metrics* d_A *and* d_B *defined on* $\mathbb{X}$ *are used to compare the distances between each pair of points* $d(\mathbf{x}_1, \mathbf{y}_1)$ *and* $d(\mathbf{x}_2, \mathbf{y}_2)$*, the probability that* d_A *and* d_B *make the same judgement about their relative distance is called* ***geometry preserving probability*** *of* d_A *and* d_B *with* f*. It is denoted by* $PG_f(d_A, d_B)$ *with mathematical description shown in (1).*

$$\begin{aligned} PG_f(d_A, d_B) = &\mathrm{P}\left[d_A(\mathbf{x}_1, \mathbf{y}_1) > d_A(\mathbf{x}_2, \mathbf{y}_2) \ \wedge \ d_B(\mathbf{x}_1, \mathbf{y}_1) > d_B(\mathbf{x}_2, \mathbf{y}_2)\right] + \\ &\mathrm{P}\left[d_A(\mathbf{x}_1, \mathbf{y}_1) < d_A(\mathbf{x}_2, \mathbf{y}_2) \ \wedge \ d_B(\mathbf{x}_1, \mathbf{y}_1) < d_B(\mathbf{x}_2, \mathbf{y}_2)\right] \end{aligned} \tag{1}$$

where $(\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, \mathbf{y}_2) \sim f$ *and* $\wedge$ *represents the logical* "and" *operator.*

By this definition, the larger $PG_f(d_A, d_B)$ is, the better the geometry is preserved from d_B to d_A. In the following parts, we will propose our multi-task metric

learning framework and then present the theoretical analysis, which shows that our method is more liable to make $PG_f(d_A, d_B)$ larger and thus can better preserve geometry. In contrast, mtLMNN focus more on propagating the absolute distances and could not leads to a large PG as ours.

2.3 Main Framework

We describe our novel multi-task metric learning framework as follows. Assume a common metric d_c is defined and the metric of each (the t-th) task d_t is enforced to be similar to d_c by a regularizer $D(d_t, d_c)$. Information contained in each metric can be propagated to others through the common metric. In case of the Mahalanobis metric, the regularizer can be also written as $D(A_t, B)$, where the matrices A_t and B correspond to the t-th task and the common one respectively. The novel framework can be formulated as

$$\min_{\{A_t\},B} \sum_t \left(L(A_t, \mathcal{S}_t) + \gamma D(A_t, B)\right) + \gamma_0 D(A_0, B) \quad \text{s.t. } A_t \in \mathcal{C}(\mathcal{S}_t), A_t \succeq \mathbf{0}, \quad (2)$$

where L is the loss function of the training samples of the t-th task $\mathcal{S}_t$ depending on the metric learning method, D is the divergence function to enforce the metric of the t-th task A_t similar to a common metric B, and $\mathcal{C}(\mathcal{S}_t)$ is the set of feasible A_t of the t-th task, which can be defined via side-information or the triplet set $\mathcal{T}_t$. The term $D(A_0, B)$ restricts B not far from a predefined metric A_0 as prior.

In this paper, we propose a framework to use the *Bregman matrix divergence* [3] as the regularizer $D(A, B)$ in (2), which is defined as

$$D_\phi(A, B) = \phi(A) - \phi(B) - \operatorname{tr}\left((\nabla\phi(B))^\top (A - B)\right),$$

where $\phi : \mathrm{SPD}(m) \to \mathbb{R}$ is a strictly convex, differentiable function.

It is easy to show that this framework includes mtLMNN as a special case by using $\phi(A) = \|A\|_{\mathrm{F}}^2$ and replacing $A_t \succeq \mathbf{0}$ with $A_t \succeq B \succeq \mathbf{0}$. However, this method has two main drawbacks: (1) The constraints $A_t \succeq B$ are unnecessarily strong for A_t to be a Mahalanobis matrix, which implies distance of any task has to be larger than the distance defined by the common part. (2) It is not appropriate to use Frobenius norm as the regularizer, since it cannot preserve the data geometry.

To overcome these drawbacks, we use the *von Neumann divergence* as the regularizer and obtain our multi-task metric learning method, where the von Neumann divergence is defined as $D_{\mathrm{vN}}(A, B) = \operatorname{tr}(A \log A - A \log B - A + B)$, where $\log A$ is the *matrix logarithm*[2] of A.

Using our method to learn a metric A that is assumed to be similar to B, it is more liable to obtain a solution with better geometry property preserved. We will detail the theoretical analysis in Section 3.

[2] If $A = V\Lambda V^\top$ is the eigendecomposition of A, the matrix logarithm is $V \log \Lambda V^\top$ where $\log \Lambda$ is the diagonal matrix containing the logarithm of eigenvalues.

An example. Now we revisit the example proposed in Fig. 2 where single-task metric learning fails to learn a good metric for any task. Fig. 2(c) and 2(d) show the data of task-1 in the metric learned using von Neumann divergence and Frobenius norm as regularizer respectively. Obviously, when Frobenius norm is used, although red points are pushed away, some testing points of green class invade into the margin again and the geometry has not been preserved. In contrast, when von Neumann divergence is used, both testing samples of red and green class are pushed outside the perimeter, which means the nice geometry property from task-2 is appropriately preserved after transferred to task-1. For task-2 shown in Fig. 2(g) and 2(h), von Neumann divergence also performs better than Frobenius norm.

Optimization. Since von Neumann divergence is jointly convex with two arguments [9], our multi-task metric learning method is jointly convex with its arguments if L is convex with A_t. This means that any convex metric learning method can be extended to our multi-task framework without losing its convexity. Therefore, it guarantees a global optimal solution and we solve it by alternating minimization method. Due to the convex, differentiable, and non-negative properties of von Neumann divergence, it is not difficult to verify the convergence of our algorithm.

Fix B and Optimize A_t. Suppose that L is convex with A_t, then the optimization is divided to T individual convex subproblems, each of which is a single-task metric learning problem with a regularizer. In this paper, we apply our multi-task framework to LMNN [10] metric learning approach which proved effective in many applications.

The subproblem for the t-th task can be solved by gradient descent method with $\frac{\partial \tilde{L}_t}{\partial A_t} = \frac{\partial L}{\partial A_t} + \gamma \frac{\partial D_{\text{vN}}}{\partial A_t} = \frac{\partial L}{\partial A_t} + \gamma(\log A_t - \log B)$. The first part is the gradient of a single-task metric learning problem, while the second part enforces A_t to be similar to a common matrix B.

Fix A_t and Optimize B. If all A_t are fixed, the variable to be optimized is B. With [1], the optimal solution of B is called the *Bregman representative* in case of matrix variables. It is straightforward to prove that Proposition 1 of [1] can be extended to the case of matrix and the minimizer is the weighted average of $\{A_t\}$ and A_0 as $B = (\gamma \sum_t A_t + \gamma_0 A_0)/(\gamma T + \gamma_0)$.

3 Theoretical Analysis

In this section, we analyze our multi-task metric learning method theoretically, showing how the von Neumann divergence encourages a larger geometry preserving probability and thus preserves geometry better. To this end, we firstly define an operator ρ called *scale extractor* to transform a metric to a vector called *scale vector*, which characterizes the important scale property of the metric. Since the scale vector is much more convenient to deal with than the metric which is a function, it provides a tool to bridge the von Neumann divergence and geometry

preserving probability. We establish such a relationship in three steps: (1) The geometry preserving probability monotonically decreases with a function of scale vectors $R(A, B)$. (2) For any orthonormal basis W and two Mahalanobis metrics d_A, d_B, the KL-divergence of $\rho_W(A)$ and $\rho_W(B)$ is bounded by the von Neumann divergence of A and B. (3) Minimizing $D_{\mathrm{KL}}(\rho_W(A), \rho_W(B))$ has the effect to minimize $R(A, B)$ and thus encourages larger geometry preserving probability $PG_f(A, B)$. These steps are discussed in detail in the following subsections.

3.1 Basic Definitions

Our motivation comes from the following fact. Given any pair of points $\forall \mathbf{x}, \mathbf{y} \in \mathbb{X}$, if two *metrics* d_A and d_B are similar, then the distances they give $d_A(\mathbf{x}, \mathbf{y})$ and $d_B(\mathbf{x}, \mathbf{y})$ are expected to be similar. It provides a way to measure the similarity between two metrics by comparing the distances they give for a certain pairs of points instead. Motivated by this, we can use a vector to characterize the properties of a metric and transform some problems from the intricate functional space $\mathcal{F}_{\mathbb{X}}$ to a much simpler vector space. Based on this idea, we propose the following definitions.

Definition 2 (Scale). *Given any metric $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+$ and a unit vector $\mathbf{w} \in \mathbb{X}$ where $\|\mathbf{w}\| = 1$, the squared distance $d^2(\mathbf{w}, \mathbf{0})$ is defined as the **scale** of d on $\mathbf{w}$.*

Since Mahalanobis metric determines a series of scales on different directions, the essential objective of metric learning is to redefine these scales with side-information so that a certain constraints are satisfied. Due to the *translation-invariant* property of Mahalanobis metric, we always translate $\mathbf{x}$ to the original and briefly denote $d_A(\mathbf{x}, \mathbf{y}) \doteq d_A(\mathbf{z})$ where $\mathbf{z} = \mathbf{x} - \mathbf{y}$ and thus the scale of d on $\mathbf{z}$ is briefly denoted as $d^2(\mathbf{z})$.

Definition 3 (Scale Extractor). *Define the operator $\rho_W : \mathcal{F}_{\mathbb{X}} \to \mathbb{R}^n$ which transforms a metric d to a vector consisting of the scales of d on a group of vectors $W_{m\times n} = [\mathbf{w}_1\ \mathbf{w}_2\ \ldots\ \mathbf{w}_n]$ as **scale extractor**:*

$$\rho_W(d) = \left[\rho_{\mathbf{w}_1}(d)\ \rho_{\mathbf{w}_2}(d) \ldots \rho_{\mathbf{w}_n}(d)\right]^\top = \left[d^2(\mathbf{w}_1)\ d^2(\mathbf{w}_2) \ldots d^2(\mathbf{w}_n)\right]^\top$$

*The vector $\rho_W(d)$ is called the **scale vector** of d on W.*

Given any W, the more similar d_A and d_B are, the more similar $\rho_W(d_A)$ and $\rho_W(d_B)$ should be. Since $\rho_W(d_A)$ and $\rho_W(d_B)$ are just real vectors, the divergence between them is much easier to estimate and has an explicit sense as metric definition for the same points. Therefore, it can be used to define $D(d_A, d_B)$ with proper W.

When estimating the divergence of two metrics $d_A, d_B \in \mathcal{F}_{\mathbb{X}}$, a natural choice of W is an orthonormal basis of $\mathbb{X}$ because they represent the scales of d_A on different directions. Then, by enforcing $\rho_W(d_A)$ and $\rho_W(d_B)$ to be similar, we can make the scales of d_A and d_B on different directions similar. As we have indicated, in metric learning problems, we hope them to be similar in the sense

of the same relative distances. In next subsections, we will show that if we choose *KL-divergence* of $\rho_W(d_A)$ and $\rho_W(d_B)$ as the regularizer, it has the effect to encourage a larger geometry preserving probability for d_A and d_B.

3.2 Enlarging $PG_f(d_A, d_B)$ by Minimizing $R(A, B)$

In this subsection, we show that the geometry preserving probability monotonically decreases with a function of scale factor vectors of two metrics, which couples the complicated defined probability with a simpler property of metric. As we have shown in Section 2.2, the geometry preserving property is mathematically measured by the geometry preserving probability PG, whose original definition is intractable, though. In following, we propose the relationship between PG and the scale vectors which correlates $PG_f(d_A, d_B)$ with the property of d_A and d_B.

For convenience of calculating PG, we first define the *geometry preserving indicator*. In the following discussion, we always denote $\mathbf{z}_i = \mathbf{x}_i - \mathbf{y}_i$ as the difference of two points.

Definition 4 (Geometry Preserving Indicator). *The* Geometry Preserving Indicator $\Psi_{A,B}(\mathbf{x}_1 - \mathbf{y}_1, \mathbf{x}_2 - \mathbf{y}_2) = \Psi_{A,B}(\mathbf{z}_1, \mathbf{z}_2)$ *is a function that takes two metrics d_A, d_B as parameters and two differences of two pairs of points as variables. We use d_A and d_B to calculate the distances of the two pairs of points and then compare which pair is relatively further. Then $\Psi = 1$ if the two metrics give the same judgement and $\Psi = 0$ otherwise. Mathematically, it is*

$$\begin{aligned}\Psi_{A,B}(\mathbf{z}_1, \mathbf{z}_2) =& \mathbb{1}\left[(d_A(\mathbf{z}_1) > d_A(\mathbf{z}_2)) \wedge (d_B(\mathbf{z}_1) > d_B(\mathbf{z}_2))\right] + \\ & \mathbb{1}\left[(d_A(\mathbf{z}_1) < d_A(\mathbf{z}_2)) \wedge (d_B(\mathbf{z}_1) < d_B(\mathbf{z}_2))\right]\end{aligned}$$

where $\mathbb{1}[\mathcal{E}]$ is the indicator function *which equals to 1 if the logical expression $\mathcal{E}$ holds and 0 otherwise.*

Noting that any $f(\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, \mathbf{y}_2)$ uniquely determines a probability density $\tilde{f}(\mathbf{x}_1 - \mathbf{y}_1, \mathbf{x}_2 - \mathbf{y}_2) = \tilde{f}(\mathbf{z}_1, \mathbf{z}_2)$ for the differences, the geometry preserving probability $PG_f(d_A, d_B)$ can be calculated as an integral on the whole space

$$PG_f(d_A, d_B) = \iint_{\mathbb{R}^m \times \mathbb{R}^m} \Psi_{A,B}(\mathbf{z}_1, \mathbf{z}_2)\tilde{f}(\mathbf{z}_1, \mathbf{z}_2)\mathrm{d}z_1^{(1)} \ldots \mathrm{d}z_1^{(m)}\mathrm{d}z_2^{(1)} \ldots \mathrm{d}z_2^{(m)} \tag{3}$$

Then we propose the theorem to couple geometry preserving probability with scales.

Theorem 1 (Geometry Preserving Theorem). *Suppose that there are two pairs of random points $\mathbf{x}_1, \mathbf{y}_1 \in \mathbb{R}^m$ and $\mathbf{x}_2, \mathbf{y}_2 \in \mathbb{R}^m$ following probability density $f(\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, \mathbf{y}_2)$. Given any $d_B \in \mathcal{F}_{\mathbb{R}^m}$, the* geometry preserving probability *$PG_f(d_A, d_B)$ is determined by $d_A \in \mathcal{F}_{\mathbb{R}^m}$ and* ***monotonically decreases*** *with*

$$R(A, B) = \iint_{\mathbb{S}^{m-1} \times \mathbb{S}^{m-1}} R_{\mathbf{w}_1, \mathbf{w}_2}(A, B)\mathrm{d}\Omega(\mathbf{w}_1)\mathrm{d}\Omega(\mathbf{w}_2) \tag{4}$$

where

$$R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)=\left|\sqrt{\frac{\rho_{\mathbf{w}_2}(A)}{\rho_{\mathbf{w}_2}(B)}}-\sqrt{\frac{\rho_{\mathbf{w}_1}(A)}{\rho_{\mathbf{w}_1}(B)}}\right|\cdot\left(\sqrt{\frac{\rho_{\mathbf{w}_1}(A)}{\rho_{\mathbf{w}_2}(B)}}+\sqrt{\frac{\rho_{\mathbf{w}_2}(A)}{\rho_{\mathbf{w}_1}(B)}}\right)^{-1}, \quad (5)$$

$\mathrm{d}\Omega(\mathbf{w}_i)$ *is the* solid angle element *corresponding to the direction of* $\mathbf{w}_i$ *which contains all the angular factors*[3] *[5], and* $\mathbb{S}^{m-1}=\{\mathbf{x}\in\mathbb{R}^m \mid \|\mathbf{x}\|=1\}$ *is the* $(m-1)$*-dimensional unit sphere in* $\mathbb{R}^m$*. The integration is calculated on* $\mathbb{S}^{m-1}$ *for both* $\mathbf{w}_1$ *and* $\mathbf{w}_2$*.*

We interpret the theorem slightly before proof. The integration (3) is taken on all the solid angle values and independent of the radius, which implies that the values of $R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$ corresponding to each pair of directions of $\mathbf{w}_1$ and $\mathbf{w}_2$ are integrated to get $R(A,B)$. Thus, if we make $R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$ smaller for each pair of directions, we can get a smaller $R(A,B)$ and a larger $PG_f(d_A,d_B)$ as we expected. What is better, this relation is independent of the distribution f. To prove the theorem, we first present a lemma.

Lemma 2. *Suppose there are two pairs of random points* $\mathbf{x}_1,\mathbf{y}_1\in\mathbb{R}^m$ *and* $\mathbf{x}_2,\mathbf{y}_2\in\mathbb{R}^m$*. For each pair, the difference* $\mathbf{z}_i=\mathbf{x}_i-\mathbf{y}_i$ *lies in a* 1-dimensional subspace $\mathbb{X}_i$ *which means there exists a unit vector* $\mathbf{w}_i\in\mathbb{X}_i$ *and a random real number* r_i *so that* $\mathbf{z}_i=r_i\mathbf{w}_i$*. Then for any Mahalanobis metrics* $d_B\in\mathcal{F}_{\mathbb{R}^m}$*, the* geometry preserving probability $PG_f(d_A,d_B)$ *is determined by* $d_A\in\mathcal{F}_{\mathbb{R}^m}$ *and* ***monotonically decreases*** *with* $R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$ *defined in (5).*

Proof. Due to the translation-invariant property of d_A and d_B, for $\forall i=1,2$, we have $d_A^2(\mathbf{x}_i,\mathbf{y}_i)=r_i^2\mathbf{w}_i^\top A\mathbf{w}_i=r_i^2\rho_{\mathbf{w}_i}(A)$, thus the squared distance d_A^2 equals to the weighted scale on $\mathbf{w}_i$ with weight r_i^2. Similarly, $d_B^2(\mathbf{x}_i,\mathbf{y}_i)=r_i^2\rho_{\mathbf{w}_i}(B)$. It is straightforward to show that

$$d_A(\mathbf{x}_1,\mathbf{y}_1)>d_A(\mathbf{x}_2,\mathbf{y}_2)\Leftrightarrow|r_1/r_2|>\sqrt{\rho_{\mathbf{w}_2}(A)/\rho_{\mathbf{w}_1}(A)} \quad (6)$$

which also holds for B. Denote $\mathbf{r}=[r_1\ r_2]^\top$ and substitute (6) into $\Psi_{A,B}$, then PG can be reformulated as a function of $\mathbf{r}$

$$PG_f(d_A,d_B)=\iint_{\mathbb{R}_+\times\mathbb{R}_+}\Psi_{A,B}(r_1\mathbf{w}_1,r_2\mathbf{w}_2)\tilde{f}(r_1,r_2)\mathrm{d}r_1\mathrm{d}r_2=\int_{S_\mathrm{I}\cup S_\mathrm{II}}\tilde{f}(\mathbf{r})\mathrm{d}\mathbf{r} \quad (7)$$

where $\tilde{f}(\mathbf{r})$ is the probability density of $\mathbf{r}$ determined by $f(\mathbf{x}_1,\mathbf{y}_1,\mathbf{x}_2,\mathbf{y}_2)$, and

$$S_\mathrm{I}=\left\{\mathbf{r}\mid|r_1/r_2|>\max\left\{\sqrt{\rho_{\mathbf{w}_2}(A)/\rho_{\mathbf{w}_1}(A)},\sqrt{\rho_{\mathbf{w}_2}(B)/\rho_{\mathbf{w}_1}(B)}\right\}\right\},$$
$$S_\mathrm{II}=\left\{\mathbf{r}\mid|r_1/r_2|<\min\left\{\sqrt{\rho_{\mathbf{w}_2}(A)/\rho_{\mathbf{w}_1}(A)},\sqrt{\rho_{\mathbf{w}_2}(B)/\rho_{\mathbf{w}_1}(B)}\right\}\right\},$$

The integral field is illustrated as the green part in Fig. 3. Since the probability density $\tilde{f}(\mathbf{r})$ is non-negative anywhere and the border corresponding to d_B is

[3] For example, for $m=2$, $\mathrm{d}\Omega(\mathbf{w}_i)=\mathrm{d}\theta$ which is independent of $\mathbf{w}_i$; for $m=3$, $\mathrm{d}\Omega(\mathbf{w}_i)=\sin\theta\mathrm{d}\theta\mathrm{d}\varphi$ where $w_i^{(1)}=\cos\theta, w_i^{(2)}=\sin\theta\cos\varphi, w_i^{(3)}=\sin\theta\sin\varphi$.

fixed, PG monotonically decreases with $|\omega|$, where ω is the angle between the two borders determined by $\rho_W(A)$ and $\rho_W(B)$. Then, we have

$$|\omega| = \left|\arctan\sqrt{\rho_{\mathbf{w}_2}(A)/\rho_{\mathbf{w}_1}(A)} - \arctan\sqrt{\rho_{\mathbf{w}_2}(B)/\rho_{\mathbf{w}_1}(B)}\right|$$
$$= \arctan\left|\frac{\sqrt{\rho_{\mathbf{w}_2}(A)/\rho_{\mathbf{w}_1}(A)} - \sqrt{\rho_{\mathbf{w}_2}(B)/\rho_{\mathbf{w}_1}(B)}}{1 + \sqrt{\rho_{\mathbf{w}_2}(A)\rho_{\mathbf{w}_2}(B)}/\sqrt{\rho_{\mathbf{w}_1}(A)\rho_{\mathbf{w}_1}(B)}}\right| = \arctan R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$$

where $R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$ is the ratio shown in (5). Since arctan is a monotony increasing function and $PG_f(d_A,d_B)$ monotonically decreases with $|\omega|$, the conclusion that $PG_f(d_A,d_B)$ monotonically decreases with $R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$ is proved. □

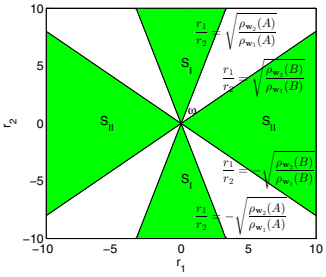
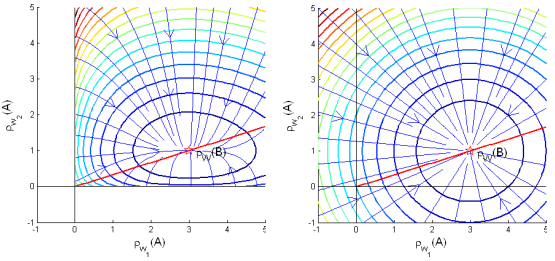


Fig. 3. Integral domain



(a) KL-divergence (b) Euclidean distance

Fig. 4. Gradient field of $D_\varphi(\rho_W(A),\rho_W(B))$.

Proof (Theorem 1). Denote $\mathbf{z}_i = r_i\mathbf{w}_i$ where $\mathbf{w}_i$ is a unit vector and r_i is the length of $\mathbf{z}_i$, then the volume element is $\mathrm{d}z_i^{(1)}\ldots\mathrm{d}z_i^{(m)} = r_i^{m-1}\mathrm{d}r_i\,\mathrm{d}\Omega(\mathbf{w}_i)$, where $\mathrm{d}\Omega(\mathbf{w}_i)$ is the solid angle corresponding to the direction $\mathbf{w}_i$.

$$\iint_{\mathbb{S}^{m-1}\times\mathbb{S}^{m-}} \iint_{\mathbb{R}_+\times\mathbb{R}_+} \Psi_{A,B}(r_1\mathbf{w}_1, r_2\mathbf{w}_2)\tilde{f}(r_1\mathbf{w}_1, r_2\mathbf{w}_2)r_1^{m-1}r_2^{m-1}\mathrm{d}r_1\mathrm{d}r_2\mathrm{d}\Omega(\mathbf{w}_1)\mathrm{d}\Omega(\mathbf{w}_2) \tag{8}$$

Note that for any fixed $\mathbf{w}_1,\mathbf{w}_2$, the inner integration is just (7) discussed in case of Lemma 2 if $\tilde{f}(r_1\mathbf{w}_1, r_2\mathbf{w}_2)r_1^{m-1}r_2^{m-1}$ is regarded as the unnormalized probability density function[4] of (r_1,r_2). Thus, replacing the inner integration of (8) with (5) and using Lemma 2, we get the conclusion that the geometry preserving probability $PG_f(d_A,d_B)$ monotonically decreases with (4). □

Remarks. Note that if d_A and d_B are learned simultaneously, PG is not guaranteed to strictly monotonically decrease with (4) because $R(A,B)$ also depends on f. However, if we have little information about f, a smaller (4) also leads to a larger PG in most cases.

[4] By the proof of Lemma 2, the conclusion also holds if f is unnormalized.

3.3 Bounding the KL-divergence with von Neumann Divergence

In this subsection, we show that by minimizing the von Neumann divergence of two Mahalanobis matrices, the KL-divergence [3] of scales on any pair of directions is minimized. This result is shown in Theorem 4 where the KL-divergence is defined as $D_{\mathrm{KL}}(\mathbf{x},\mathbf{y}) = \sum_i x_i(\log x_i - \log y_i) - x_i + y_i$.

The main result Theorem 4 is supported by Lemma 3, a result very similar to that in quantum information [7]. We have to omit the detailed proof due to the limit of space and present only the results. We found that it can be proved in the similar way as [7].

Lemma 3. *For any trace preserving map [7] Φ, given by $\Phi(A) = \sum_{i=1}^{n} V_i A V_i^\top$ and $\sum_{i=1}^{n} V_i^\top V_i = \mathbf{I_m}$, we have that $D_{\mathrm{KL}}\left(\Phi(A), \Phi(B)\right) \leq D_{\mathrm{vN}}(A,B)$.*

Theorem 4. *Suppose $d_A, d_B \in \mathcal{F}_{\mathbb{R}^m}$ are two Mahalanobis metrics defined on $\mathbb{R}^m$, then for any orthonormal basis $W = [\mathbf{w}_1 \ \ldots \ \mathbf{w}_m]$ in $\mathbb{R}^m$, the* KL-divergence of their scale vectors $\rho_W(A)$ *and* $\rho_W(B)$ *is bounded by the* von Neumann divergence of their Mahalanobis matrices A *and* B*:* $D_{\mathrm{KL}}(\rho_W(A), \rho_W(B)) \leq D_{\mathrm{vN}}(A,B)$.

Proof. For any orthonormal basis $W = [\mathbf{w}_1 \ \ldots \ \mathbf{w}_m]$, we have

$$\begin{aligned} D_{\mathrm{KL}}\left(\rho_W(A), \rho_W(B)\right) &= \sum_i D_{\mathrm{KL}}(\mathbf{w}_i^\top A \mathbf{w}_i, \mathbf{w}_i^\top B \mathbf{w}_i) \\ &= \sum_{i,j} (\mathbf{w}_i^\top \mathbf{w}_j)^2 D_{\mathrm{KL}}(\mathbf{w}_i^\top A \mathbf{w}_i, \mathbf{w}_j^\top B \mathbf{w}_j) \\ &= D_{\mathrm{vN}}(\sum_i W_i A W_i^\top, \sum_i W_i B W_i^\top) \leq D_\phi(A,B) \end{aligned}$$

where $W_i = \mathbf{w}_i \mathbf{w}_i^\top$. The third equality is the decomposition of Bregman matrix divergence [3] and the last inequality results from Lemma 3. □

Using Theorem 4, it is easy to show that minimizing $D_{\mathrm{vN}}(A,B)$ has the effect to minimize $D_{\mathrm{KL}}(\rho_{\mathbf{w}}(A), \rho_{\mathbf{w}}(B))$ on any direction $\mathbf{w}$. Interestingly, when $D(A,B) = \|A-B\|_F^2$ is used, a similar result can be attained using simple matrix calculation. We propose it in Theorem 5 and omit the proof.

Theorem 5. *Suppose $d_A, d_B \in \mathcal{F}_{\mathbb{R}^m}$ are two Mahalanobis metrics defined on $\mathbb{R}^m$, then for any orthonormal basis $W = [\mathbf{w}_1 \ \ldots \ \mathbf{w}_m]$ in $\mathbb{R}^m$, the* squared Euclidean distance of their scales $\rho_W(A)$ *and* $\rho_W(B)$ *is bounded by the* squared Frobenius norm of the difference of their Mahalanobis matrices A *and* B*:* $\|\rho_W(A) - \rho_W(B)\|^2 \leq \|A-B\|_F^2$.

In the language of Bregman divergence, the results of Theorem 4 and Theorem 5 can be uniformly formulated as $D_\varphi(\rho_W(A), \rho_W(B)) \leq D_\phi(A,B)$, where D_φ and D_ϕ are *Bregman divergence* and *Bregman matrix divergence* with the same seed function ($\phi = \varphi \circ \lambda$). However, this result cannot be straightforwardly extended to other Bregman divergences.

3.4 Minimizing $R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$ by Minimizing $D_{\mathrm{KL}}(\rho_W(A),\rho_W(B))$

As we have proved that minimizing $D_{\mathrm{vN}}(A,B)$ is to minimize $D_{\mathrm{KL}}(\rho_W(A),\rho_W(B))$ for any W, in this subsection, we then show that it furthermore encourages a smaller $R(A,B)$ in (4) and thus a larger $PG(A,B)$ by Theorem 1.

Supposing there is a metric learning problem[5] $\min_A L(A,\mathcal{S})$ whose optimal solution is $\tilde{A}$, we have $\nabla_A L\mid_{\tilde{A}}=0$. If there exists a related task with optimal solution B and we would like to propagate the information embedded in B to A, the optimization formula becomes $\min_A L(A,\mathcal{S})+\gamma D_{\mathrm{vN}}(A,B)$ where a new loss function is added to L and the optimal solution should move to another point with a smaller loss. Obviously, it always moves towards the negative gradient direction where the loss is smaller.

Here we study how $\rho_W(A)$ is effected by the regularizer for any given W. As we have shown, when $D_{\mathrm{vN}}(A,B)$ is added to loss function, it aims to minimize $D_{\mathrm{KL}}(\rho_W(A),\rho_W(B))$ and thus $\rho_W(A)$ is more liable to move towards $-\nabla D_{\mathrm{KL}}(\rho_W(A),\rho_W(B))$. The gradient of D_{KL} with respect to $\rho_W(A)$ is

$$\nabla D_{\mathrm{KL}}=\left[\log(\rho_{\mathbf{w}_1}(A)/\rho_{\mathbf{w}_1}(B))\ \ldots\ \log(\rho_{\mathbf{w}_n}(A)/\rho_{\mathbf{w}_n}(B))\right]^\top.$$

By its formulation, the gradient on each direction $\mathbf{w}_i$ is proportional to the logarithm of the ratio of scales on $\mathbf{w}_i$. This means that the regularizer always enforces the component $\rho_{\mathbf{w}_i}(A)$ with larger $\rho_{\mathbf{w}_i}(A)/\rho_{\mathbf{w}_i}(B)$ decreases more quickly, which *encourages the ratios of scales* $\rho_{\mathbf{w}_i}(A)/\rho_{\mathbf{w}_i}(B)$ *on different* $\mathbf{w}_i$ *equal*.

Noting that the numerator of $R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$ in (5) is the absolute value of difference of the ratios of scales on two directions, encouraging $\rho_{\mathbf{w}_1}(A)/\rho_{\mathbf{w}_1}(B)=\rho_{\mathbf{w}_2}(A)/\rho_{\mathbf{w}_2}(B)$ to be equal is to minimize $R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$. Thus the main conclusion of this subsection can be proposed as

Proposition 1. *For any n unit vectors $W=[\mathbf{w}_1\ \ldots\ \mathbf{w}_n]\in\mathbb{R}^{m\times n}$, the regularizer $\min_{\rho_W(A)} D_{\mathrm{KL}}(\rho_W(A),\rho_W(B))$ encourages the solution to make $R_{\mathbf{w}_i,\mathbf{w}_j}(A,B)$ smaller for $\forall i,j$.*

In contrast, if $D(A,B)=\|A-B\|_F^2$ is used, the equivalent regularizer is $\|\rho_W(A)-\rho_W(B)\|^2$. It *encourages the differences of scales* $(\rho_{\mathbf{w}_i}(A_t)-\rho_{\mathbf{w}_i}(B))$ *on different* $\mathbf{w}_i$ *equal*, which is not beneficial to minimizing $R_{\mathbf{w}_i,\mathbf{w}_j}(A,B)$.

This phenomenon is illustrated with the contour and gradient field in Fig. 4 where the red line represents the points with the same ratio of scales. The concentric circles are contours of $D_\varphi(\rho_W(A),\rho_W(B))$ and the radial lines are field lines of its negative gradient where the tangent direction at any point of the line indicates $-\nabla_{\rho_W(A)}D_\varphi(\rho_W(A),\rho_W(B))$. Minimizing $D_\varphi(\rho_W(A),\rho_W(B))$ with respect to $\rho_W(A)$ will make the solution move along the gradient field lines because it directs to the steepest descendent direction. From this figure, we see that the field lines in Fig. 4(a) are more liable to go towards the red line, which makes the solution of $\rho_{\mathbf{w}_1}(A)/\rho_{\mathbf{w}_1}(B)$ more similar to $\rho_{\mathbf{w}_2}(A)/\rho_{\mathbf{w}_2}(B)$.

[5] The constraints can be reformulated into loss function using Lagrangian multiplier.

3.5 Summary

As a short summary of previous theoretical analysis, we have Proposition 2 for our multi-task metric learning framework.

Proposition 2 (Geometry Preserving with von Neumann divergence). *When the von Neumann divergence is minimized, the KL-divergence of the scales on different directions is minimized. This makes a smaller* $R_{\mathbf{w}_1,\mathbf{w}_2}(A,B)$ *for any pair of directions and thus a smaller* $R(A,B)$*, further leading to a larger* $PG_f(d_A,d_B)$ *by Theorem 1. In short, von Neumann divergence* $D_{\text{vN}}(A,B)$ *encourages a larger* $PG_f(A,B)$ *and can thus better propagate the side-information about relative distance.*

4 Experiments

In this section, we conduct a series of experiments to validate the advantages of our proposed approach. In our experiments, we choose LMNN [10] as the metric learning algorithm for all methods which determines the loss function L in (2). For brevity, we call our proposed multi-task metric learning with von Neumann divergence as *mt-von*, while the method proposed in [8] is written in short as *mt-Frob* (also called mtLMNN). We compare them with three baseline methods: the *Euclidean* metric, the single-task metric learning (in short *stLMNN*) and the uniform task metric learning (in short *utLMNN*). stLMNN means that a metric is learned for each task independently, while utLMNN puts the samples of all tasks together and train a uniform metric for all tasks.

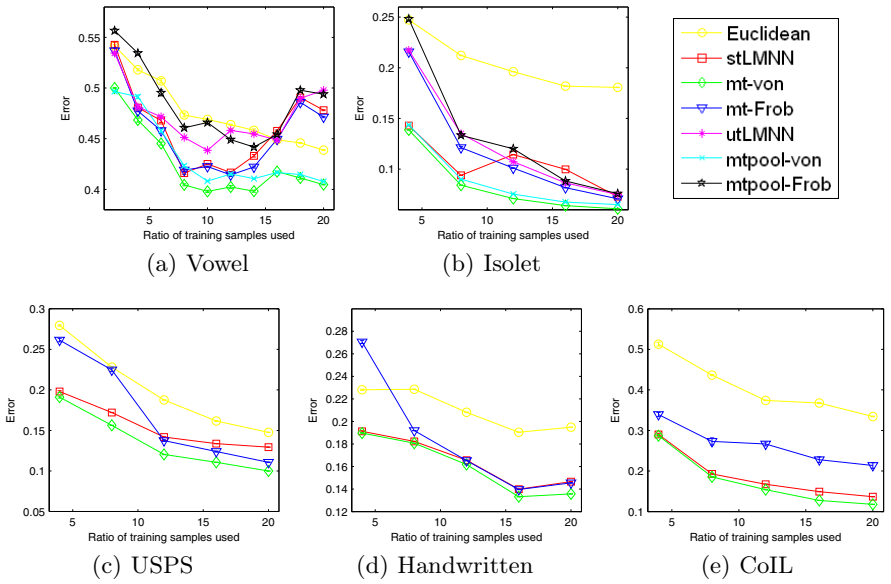
We learn a specific metric using different methods. According to the distances calculated based on the learned metric, we use 1-Nearest Neighbor as the final classifier to predict the label of a new test sample. If all tasks share a common label space, which is referred as the *label-compatible* scenario [8], we also evaluate with the *pooled training sets* [8] at the classification phase. This special classification setting is called *mtpool-von* or *mtpool-Frob*, depending on the regularizer. We also report the performance of nearest neighbor using the Euclidean distance (in short *Euclidean*) as the baseline. We tune the hyper-parameters involved in LMNN by cross validation.

We evaluate the above mentioned methods on five real data sets obtained from very different disciplines. (1). **Handwritten Letter Classification** dataset[6] consists of 8 binary handwritten letter classification problems. Each classification problem is regarded as one task. Some randomly selected samples are used to train a metric while the remaining for test. (2). **USPS digit** dataset[7] consists of 7,291 16×16 grayscale images of digits $0 \sim 9$. For each digit, we can get a two-class classification task in which the samples of this digit represent the positive patterns and the others negative patterns. (3). **Isolet** dataset[8] was collected from 150 speakers uttering all characters in the English alphabet twice. The task is

[6] http://multitask.cs.berkeley.edu/

[7] http://www-i6.informatik.rwth-aachen.de/~keysers/usps.html

[8] Available from UCI Machine Learning Repository.



(a) Vowel (b) Isolet

(c) USPS (d) Handwritten (e) CoIL

Fig. 5. Experiment results on five datasets

to classify the letter to be uttered. The speakers are grouped into 5 smaller sets of similar speakers and this makes the problem naturally be suitable for multi-task learning. Each subset is treated as a task and they are trained jointly. (4). **Insurance Company** (CoIL) Benchmark dataset[9] contains information on customers of an insurance company. The data consist of 86 variables. We select out the $68 \sim 73$-th variables as categorical features and predict their values with other features. (5). **Multi-speaker Vowel Classification** dataset[10] consists of 11 vowels uttered by 15 speakers of British English. We used the data of 1-8 (9-15) speakers as the training (testing) set. In both of them, speakers are divided into two subgroups according to their gender. It is reasonable because men pronounce in a different style with women. For this dataset, we treat each subgroup as a task.

For the first 4 datasets, we randomly choose a certain number of samples as the training set and leave the remaining samples as the test set. For the Multi-speaker Vowel dataset, we randomly select a number of samples from the 1-8 speakers as the training samples, and consider all the samples from the 9-15 speakers as the test set. In each experiment, we vary the number of training samples in each class from 4 to 20 and repeat the evaluations 10 times. The average error rates over all the tasks and the 10 times evaluations are reported in Fig. 5 as the final results. Note that, similar to [8], the five datasets are categorized into *label-compatible* and *label-incompatible* according to whether all

[9] http://kdd.ics.uci.edu/databases/tic/tic.html

[10] Available from UCI Machine Learning Repository.

tasks share a common label space. For label-compatible datasets, we compare all approaches mentioned above; for label-incompatible datasets, since tasks have different label spaces and $\bigcup \mathcal{S}_\tau$ is meaningless, the utLMNN, mtpool-von, and mtpool-Frob are not evaluated.

Observed from the experimental results, our proposed multi-task metric learning method performs the best across all the data sets whatever the number of training samples are used. This clearly demonstrates the superiority of our proposed multi-task framework. In particular, the geometry preserving mt-von method demonstrated significantly better performance against mt-Frob or mtLMNN consistently in all the cases. This clearly validates that the performance can be improved by preserving relative distances. For the label-compatible datasets, we see that in most cases, the performance is better if only the training samples in the task are used as the prototype of k-NN classifier. This once again demonstrates the advantages of our proposed method.

5 Conclusion

In this paper, we propose a novel multi-task metric learning framework using von Neumann divergence. On one hand, the novel regularized approach extends previous methods from the vector regularization to a general matrix regularization framework; on the other hand and more importantly, by exploiting von Neumann divergence as the regularizer, the new multi-task metric learning has the capability to well preserve the data geometry. This leads to more appropriate propagation of side-information among tasks and proves very important for further improving the performance. We propose the concept of *geometry preserving probability (PG)* and justify our framework with a series of theoretical analysis. Furthermore, our formulation is jointly convex and the global optimal solution can be guaranteed. A series of experiments verify that our proposed algorithm can significantly outperform the current methods.

Acknowledgements. This work has been supported in part by the National Basic Research Program of China (973 Program) Grant 2012CB316301, the National Natural Science Foundation of China (NSFC) Grants 61075052 and 60825301, and Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation.

References

1. Banerjee, A., Merugu, S., Dhillon, I.S., Ghosh, J.: Clustering with bregman divergences. Journal of Machine Learning Research 6, 1705–1749 (2005)
2. Burago, D., Burago, Y., Ivanov, S.: A Course in Metric Geometry. American Mathematical Society (June 2001)
3. Dhillon, I.S., Tropp, J.A.: Matrix nearness problems with bregman divergences. SIAM Journal on Matrix Analysis and Applications 29, 1120–1146 (2008)

4. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning multiple tasks with kernel methods. Journal of Machine Learning Research 6, 615–637 (2005)
5. Haber, H.E.: The volume and surface area of ann-dimensional hypersphere (2011), http://scipp.ucsc.edu/~haber/ph116A/volume_11.pdf
6. Huang, K., Ying, Y., Campbell, C.: Generalized sparse metric learning with relative comparisons. Knowledge and Information Systems 28(1), 25–45 (2011)
7. Lindblad, G.: Completely positive maps and entropy inequalities. Commun. Math. Phys. 40(2), 147–151 (1975)
8. Parameswaran, S., Weinberger, K.: Large margin multi-task metric learning. In: Advances in Neural Information Processing Systems 23, pp. 1867–1875 (2010)
9. Tropp, J.A.: From joint convexity of quantum relative entropy to a concavity theorem of lieb. Proceedings of the American Mathematical Society 140, 1757–1760 (2012)
10. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research 10, 207–244 (2009)
11. Yang, P., Huang, K., Liu, C.L.: A multi-task framework for metric learning with common subspace. Neural Computing and Applications, 1–11 (2012)
12. Zhang, Y., Yeung, D.Y.: Transfer metric learning by learning task relationships. In: Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2010)

Learning and Inference in Probabilistic Classifier Chains with Beam Search

Abhishek Kumar[1,⋆], Shankar Vembu[2,⋆], Aditya Krishna Menon[1], and Charles Elkan[1]

[1] Department of Computer Science, UC San Diego, USA
{abhishek,akmenon,elkan}@ucsd.edu
[2] Donnelly Centre for Cellular and Biomolecular Research, University of Toronto, Canada
shankar.vembu@utoronto.ca

Abstract. Multilabel learning is an extension of binary classification that is both challenging and practically important. Recently, a method for multilabel learning called *probabilistic classifier chains* (PCCs) was proposed with numerous appealing properties, such as conceptual simplicity, flexibility, and theoretical justification. However, PCCs suffer from the *computational* issue of having inference that is exponential in the number of tags, and the *practical* issue of being sensitive to the suitable ordering of the tags while training. In this paper, we show how the classical technique of *beam search* may be used to solve both these problems. Specifically, we show how to use beam search to perform tractable test time inference, and how to integrate beam search with training to determine a suitable tag ordering. Experimental results on a range of multilabel datasets show that these proposed changes dramatically extend the practical viability of PCCs.

1 Introduction

In the classical supervised learning task of binary classification, our goal is to learn some model that, given an input $x \in \mathcal{X}$, returns a single binary prediction $y \in \{0, 1\}$. This value y is considered to be a *label* of the example x, denoting whether it possesses some characteristic, or not. For example, x may represent an image by its pixel values, and y may denote whether or not the image contains a face. Multilabel learning is an extension of binary classification where the goal is to return *multiple* binary predictions, or equivalently a vector $y \in \{0, 1\}^K$. The label y now measures multiple characteristics of the example x, each of which we call a *tag*. For example, x may represent an image as before, and y could denote whether K specific people's faces appear in the image.

The recently proposed probabilistic classifier chains [1,2] (PCCs) are an attractive solution for multilabel classification for several reasons. First, it is based on a reduction of multilabel to binary classification, which allows us to leverage existing research on designing scalable and powerful binary classifiers. Second,

⋆ Contributed equally.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 665–680, 2012.

it is a principled probabilistic model, and there is a theoretical understanding of how it may be used to produce Bayes optimal predictions for a variety of loss functions [1]. Third, it is computationally inexpensive to train (unlike e.g. structured prediction methods [3], which involve inference during training). Fourth, it is trained on the original label space without any prior transformations [4,5,6,7], which is important in certain settings and applications.

Despite the above positive characteristics, the current formulation of PCCs suffers from at least a couple of drawbacks. First, on the *computational* side, they are only applicable to multilabel data sets with a few number of tags. This is because to use a PCC at test time for an example with K possible tags, we need to evaluate all 2^K possible labellings, and pick the highest scoring one. This becomes quickly infeasible as K increases. Second, on the *performance* side, their accuracy depends on a pre-specified ordering of the tags. Different orderings result in solutions of different accuracy, and so a natural question is whether one can determine the ordering that yields the best performance. As with the previous issue, the current understanding of PCCs requires either picking a random ordering, or trying all $K!$ possibilities.

In this paper, we propose to address these shortcomings with PCCs using *beam search*, a classical AI search technique. In particular, we propose to use beam search to perform inference on PCCs at test time, changing the runtime from $O(2^K)$ to $O(bK)$, where b is a tunable beam width. As we shall demonstrate, in practice a beam size $b \ll 2^K$ achieves good performance. We also present an algorithm that integrates the search for the best ordering of tags with the learning algorithm. To avoid the burden of training a classifier for each ordering, we use *kernel target alignment* [8] to score the viability of a given ordering. Finally, we propose a richer feature representation for learning individual tag models than that is used in existing PCC solutions [1,2]. Experimental results on a range of multilabel data sets show that our scheme is able to improve on PCC, and extend its applicability to data sets with a large number of tags.

This paper is organized as follows. First, in Section 2, we analyze PCCs in detail and highlight some of the challenges in using them. Next, in Section 3, we show how one may use beam search to speed up test time inference of the method. In Section 4, we then show how beam search may be integrated during the learning phase to determine the tag ordering. Finally, we present a range of experimental results in Section 5.

2 Multilabel Learning and Probabilistic Classifier Chains

2.1 Multilabel Learning

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the input space and $\mathcal{Y} = \{0,1\}^K$ be the label output space defined over a fixed set of K *tags*. Given a set of m training samples $\mathcal{T} = \{(x^{(i)}, y^{(i)})\}_{i=1}^m$ where $(x^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$, the goal of a multilabel classification algorithm is to learn a mapping $f : \mathcal{X} \to \mathcal{Y}$. We will use the notation $y_k^{(i)}$ to denote the kth tag of the ith example.

The naïve solution to the multilabel learning problem is to decompose it into K independent binary classification problems, one for each tag y_k. This method is known as *binary relevance*. In principle, this method is in fact optimal for certain loss functions, such as the Hamming and the ranking loss [1]. However, in practical situations where training data is limited, and for loss functions that take into account consistency of the entire tag sequence, it is intuitively necessary to exploit correlations between tags to make better predictions. This has motivated a slew of multilabel learning methods (see [9,10,11] for surveys).

Recently, Read *et al.* [2] proposed a simple decomposition method called the *classifier chain* (CC) that appears similar to binary relevance, but is able to exploit tag correlations. As with binary relevance, the idea is to train K separate models, one for each tag. The difference is that in the model for tag k, we use as input features not only the data point x but also the $(k-1)$ tags, $y_1, y_2, \ldots, y_{k-1}$, previously modeled. We thus attempt to use any relevant information in the previous tags to help simplify our model. Our focus in this paper is the related PCC method [1], which generalizes this scheme through a probabilistic framework.

2.2 The PCC Model for Multilabel Learning

A probabilistic classifier chain [1] tries to estimate the conditional distribution $p(y \mid x)$ using the chain rule of probabilities:

$$p(y \mid x) = p(y_{\pi(1)} \mid x) \prod_{k=2}^{K} p(y_{\pi(k)} \mid x, y_{\pi(1)}, \ldots, y_{\pi(k-1)}) \ ,$$

where $\pi(\cdot)$ is some fixed permutation/ordering of tags. Thus, learning a multilabel classifier is reduced to learning K independent probabilistic binary classifiers. These independent base classifiers may be, for example, logistic regression models with a specialized feature representation:

$$p(y_{\pi(k)} \mid x, y_{\pi(1)}, \ldots, y_{\pi(k-1)}; \theta) \propto \exp(\langle \theta_{\pi(k)}, \phi_{\pi(k)}(x, y) \rangle), \quad \forall k \in \{2, \ldots, K\}.$$

In [1], the choice $\phi_k(x, y) = x \oplus (y_1, \ldots, y_{k-1})$ was used, where $a \oplus b$ is the concatenation of the vectors a and b, so that $\theta_k \in \mathbb{R}^{d+k-1}$. In total, this means we need to learn $\mathbb{R}^{dK+K(K-1)/2}$ parameters, as opposed to R^{dK} parameters with binary relevance. Suppose we write $\theta_k = \left[w_k; v_k\right]$, where $w_k \in \mathbb{R}^d$ and $v_k \in \mathbb{R}^{k-1}$. Then, it may be verified that the joint probability model with a logistic regression base learner is

$$p(y \mid x; \theta) = \frac{\exp\left(y^T W x + y^T V y\right)}{\prod_{k=1}^{K} (1 + \exp\left((Wx + Vy)_k\right))}, \tag{1}$$

where we have $W = \left[w_1 \ldots w_K\right]$ and $V = \left[v_1 \ldots v_K\right]$. The matrix V is lower-triangular, since we first model $y_{\pi(1)}$, then $y_{\pi(2)}$, *et cetera*. By contrasting this to the joint model assumed for binary relevance,

$$p(y \mid x; \theta) = \frac{\exp\left(y^T W x\right)}{\prod_{k=1}^{K} (1 + \exp((Wx)_k))} \ ,$$

we see that the key difference in the joint probability model (1) is the tag correlation term $y^T V y$.

2.3 Advantages of Using PCCs

PCCs have a number of attractive properties as a multilabel classification method.

(i) It is based on a reduction from multilabel to binary classification, which allows us to leverage existing research on designing scalable and powerful binary classifiers. Compared to the original CC method [2], which also uses decomposition, the key difference in this regard is that the decomposition is probabilistically motivated. Also, unlike CC, PCC does not use the model's *predictions* of the past tags during test time inference.

(ii) It is a principled probabilistic model with a theoretical understanding of how it may be used to produce Bayes optimal predictions for a variety of loss functions [1]. This is in contrast to several multilabel learning methods, where the statistical consistency of the algorithm is unclear. Further, the probabilistic underpinning gives a clear idea on how to modify the algorithm. For example, as we shall see later, the probabilistic setup allows one to design inference methods that are more efficient than the greedy inference described in [2].

(iii) Being a decomposition method, it is computationally inexpensive to train, requiring only marginally more effort than the binary relevance baseline. This is unlike e.g. structured prediction methods [3] which involve inference during training.

(iv) It is trained on the original label space without any prior transformations [4,5,6,7]. This is conceptually appealing and makes modifications much simpler. As an example, suppose we want to address the issue of class imbalance at the tag level. One way to do this is to appropriately modify the inputs to the models for each $p(y_k \mid x, y_1, \ldots, y_{k-1})$ by applying cost-sensitive weighting. By contrast, in transformation methods, since we lose the relationship to the original label space, it is not clear how modifications in the transformed space affect those in the original space.

2.4 Challenges with Using PCCs

Thus far, we have not discussed two crucial questions in using PCCs: how to train them, and how to apply them at test time. At *training time*, one may maximize the log-likelihood of the given training set, which decomposes into K distinct optimizations for each tag:

$$\begin{aligned}\mathcal{L}(\theta; \pi(\cdot)) &= \sum_{i=1}^{m} \log p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \sum_{i=1}^{m} \left[\log p(y^{(i)}_{\pi(1)} \mid x^{(i)}; \theta) + \sum_{k=2}^{K} \log p(y^{(i)}_{\pi(k)} \mid x^{(i)}, y^{(i)}_{\pi(1)}, \ldots, y^{(i)}_{\pi(k-1)}; \theta) \right]. \end{aligned} \tag{2}$$

The above hides a subtle issue: while in theory the chain rule applies regardless of the ordering of the tags, in practice, the ordering can make a big difference. The reason is that our model for each individual $p(y_{\pi(k)} \mid x, y_{\pi(1)}, \ldots, y_{\pi(k-1)})$ may be misspecified, in which case some orderings will be better modeled than others. Therefore, we can expect different solutions based on the choice of $\pi(\cdot)$. This prompts the natural question of what the *best* ordering $\pi(\cdot)$ is, in the sense of resulting in the highest possible value of $\mathcal{L}(\theta)$. It may seem that one should order the tags in order of "difficulty", but this may not be optimal: for example, a tag that is difficult to model may make subsequent tags considerably easier to model. Thus, some principled algorithmic solution is necessary.

At *test time*, the problem becomes one of estimating, for a given feature vector x, the most likely set of tags under the learned parameters $\hat{\theta}$:

$$\hat{y} = \operatorname*{argmax}_{y \in \{0,1\}^K} p(y \mid x; \hat{\theta}) \ .$$

This inference is unfortunately computationally intractable. The proposal in [1] is to simply perform brute-force enumeration of all possible labels.

To summarize, we see that there are two main issues with using PCCs in practice. On the *computational* side, the method proposed for test time inference in [1] requires that we enumerate all 2^K possible candidate labellings, and evaluate them. Indeed, existing applications of PCCs have been restricted to data sets with relatively few number of tags. A general purpose multilabel method should of course handle a large number of tags. On the *accuracy* side, the choice of ordering the tags while training can make a difference in generalization performance. One might hope to do significantly better than just a random ordering. While Dembczyński *et al.* [1] proposed taking several random orderings to create an ensemble of PCCs, we would like a more principled procedure, that searches more intelligently.

We note that there are related schemes that deal with the above problems for PCCs. An inference algorithm was proposed in [12] which makes assumptions on the joint probability distribution of labels to guarantee polynomial-time convergence of the algorithm. However, it does not address the problem of learning tag orderings. Our algorithm based on beam search is generic in the sense that it is possible to accommodate a variety of scoring functions into the search algorithm thereby allowing us to solve both the inference problem and the problem of learning tag orderings. In [13], an algorithm is proposed to learn an undirected network of dependencies between the tags, which is intractable in general, and therefore approximates the structure learning problem using the Chow-Liu algorithm to learn a tree dependency structure between tags. However, this tree structure is unlikely to represent many real-world scenarios and it is an empirical question whether such an approximation is good or not.

3 Label Inference Using Beam Search

Recall that the inference problem in PCC is $\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} \ p(y \mid x; \hat{\theta})$, i.e., we wish to find the maximum scoring label vector. Assuming our probability

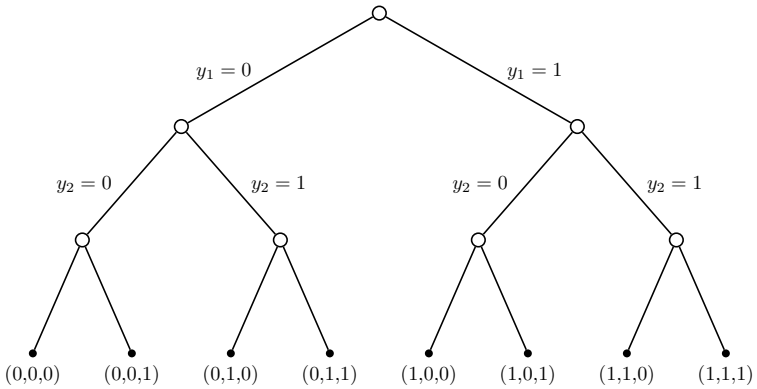


Fig. 1. Binary tree used in beam search for inference with $K = 3$ tags

model is correctly specified, the resulting solution will give the Bayes optimal prediction for subset 0/1 loss [1]. This inference task is equivalent to finding the optimal path in a rooted, complete binary tree of height K, where each internal vertex at level k denotes a possible *partial label vector* of length k, so that the leaf nodes represent all possible 2^K label vectors (see Figure 1). Thus, the inference problem is one of finding the optimal path from the root to one of the leaves in this binary tree, where the score of a vertex v at level k with a corresponding partial label $y^{(v)}$ is equal to the partial probability:

$$s_k(v; \hat{\theta}) = p(y_1^{(v)} \mid x; \hat{\theta}) \cdot \prod_{j=2}^{k} p(y_j^{(v)} \mid x, y_1^{(v)}, \ldots, y_{j-1}^{(v)}; \hat{\theta}) \ , \tag{3}$$

which can be computed recursively. The inference algorithm in the original classifier chain [2] greedily searches for the optimal path by deciding at each level of the tree whether to traverse in the left or the right direction. However, this may not result in finding the optimal label vector [1].

We propose an inference algorithm using *beam search* [14], which is a heuristic search technique. A*[15] and similar search algorithms could also be used as more sophisticated alternatives to beam search. The basic idea is that we will keep b candidate solutions at each level of the tree, where b is a user-defined parameter known as the beam width, which represent the best partial solutions seen thus far. We then explore the tree in a breadth-first fashion using these solutions. Greedy search is recovered for the case of $b = 1$.

Our inference procedure for PCCs is described in Algorithm 1. At each level of the tree, we maintain a list of best-scoring candidate vertices of size at most b, where b is the beam width. We traverse down the tree by considering the children of only those vertices that are in this list, sort them in increasing order of their partial probabilities (3), and prune all the vertices that are not in the top-b list.

Algorithm 1. Inference using beam search.

Input: Query point x, learned model parameters $\hat{\theta}$, beam width b
Output: Estimate $\hat{y}$ for $\operatorname{argmax}_y p(y \mid x; \hat{\theta})$

$B^{(0)} = \{(1, 0)\}$ {initialize beam}
for $j = 1 \ldots K$ **do**
  $B^{(j)} = \{\}$
  for (parentTags, parentScores) $\in B^{(j-1)}$ **do**
    for $z \in \{0, 1\}$ **do**
      if $p(y_j = z \mid x, \text{parentTags}; \hat{\theta}) > \min\{v : (\cdot, v) \in B^{(j)}\}$ **then**
        $B^{(j)} \leftarrow B^{(j)} \cup (\text{parentTags} \cup \{z\}, p(y_j = z \mid x, \text{parentTags}; \hat{\theta}))$
        $B^{(j)} \leftarrow \text{Top-}b(B^{(j)})$
      end if
    end for
  end for
end for
$\hat{v} = \operatorname{argmax}_v \{v : (\cdot, v) \in B^{(K)}\}$ {highest score}
return $\hat{y} : (\hat{y}, \hat{v}) \in B^{(K)}$

The greedy inference algorithm used in classifier chain [2] is recovered as a special case with beam width $b = 1$ and performing inference by exhaustively enumerating all possible labels is equivalent to doing beam search with $b = \infty$. Thus, by tuning b, we can control the tradeoff between computation time and accuracy of our method. The hope is that for real-world multilabel datasets, we can use a relatively small value of b and get performance that is significantly better than the greedy approach, and commensurate with exhaustive enumeration. This is a question that we will answer empirically in Section 5.

4 Learning to Order Tags

Recall that training a PCC involves picking a particular ordering $\pi(\cdot)$ of the tags, based on which we apply the chain rule decomposition. The tag ordering problem is to find the best $\pi(\cdot)$ in the sense of yielding the maximum log-likelihood $\mathcal{L}(\theta; \pi(\cdot))$ in Equation 2. It is easy to see that if each individual tag model is misspecified, this quantity varies based on the choice of the permutation $\pi(\cdot)$. Even if the model is correctly specified, the optimal solution may vary due to finite sample effects; for example, suppose that a tag y_k is extremely rare; then, on any finite sample, we may seriously misestimate $p(y_k = 1 \mid x)$, even if in the infinite sample case we will discover the correct probability.

Intuitively, one may expect the optimal ordering to progressively involve picking the easiest tag to model given the previously picked tags. But it may alternately be the case that given a "difficult" tag, subsequent tags are easy to model. A basic question then is to how to determine a suitable tag ordering without resorting to heuristics, or performing exhaustive enumeration over all $K!$ possible orderings.

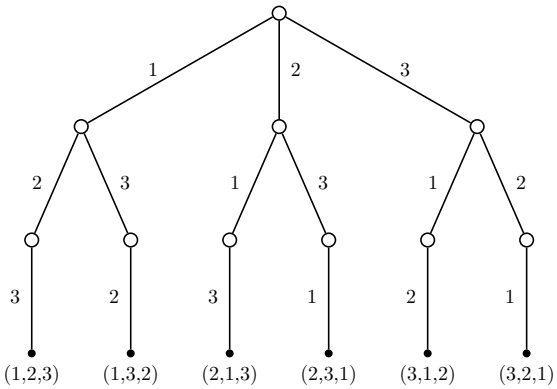


Fig. 2. Example of ordering tree for $K = 3$ tags

We propose to use beam search to solve the problem of determining a suitable tag ordering. We do so by casting it as a search problem over a tree. Instead of a complete binary tree used in the inference algorithm, for the ordering problem we have a tree of height K, where every vertex at level t has $(K - t)$ children, as shown in Figure 2. Given such a tree, our goal is again to find the optimal path from the root to one of the leaf vertices.

Our procedure to learn tag orderings for PCC is described in Algorithm 2. Similar to the beam search algorithm used for inference, we use a beam of fixed width b, maintain a list of best-scoring candidate vertices of size at most b and prune all the vertices that are not in the top-b list. We now need to determine the scoring function used to prune the vertices. One possible scoring function is the validation error of classifier, i.e., for every candidate vertex in the tree, we train a (partial) PCC. More specifically, the score of a vertex v at level t is given by:

$$s_t(v; \hat{\theta}) = \sum_{(x,y) \in \mathcal{V}} \log p(y_{\pi_v(t)} \mid x, y_{\pi_v(1)}, \ldots, y_{\pi_v(t-1)}; \hat{\theta}) \ ,$$

where $\hat{\theta}$ are the parameters of the (partial) PCC that is being evaluated on a validation set of examples $\mathcal{V}$, and the partial tag ordering specified by $\pi_v(\cdot)$ is the directed path from the root to the vertex v. However, this results in training a (partial) PCC at every vertex of the tree which can be prohibitively expensive. As a computationally cheaper alternative, we propose to instead use *kernel target alignment* (KTA) [8] as a measure to score vertices. We want to measure to what extent similar training examples agree on a single given binary tag. Let $y \in \{0, 1\}^m$ be a vector containing the value of this tag for each of the m training examples. The matrix $yy^\top$ is a kernel matrix based on this tag. Let $\mathcal{K}$ be the kernel matrix based on the feature vector representing each of the m examples. Let $\langle A, B \rangle_F = \sum_{ij} A_{ij} B_{ij}$ denote the Frobenius inner product between two matrices A and B. The kernel target alignment between $\mathcal{K}$ and $yy^\top$ is

Algorithm 2. Learning to order tags using beam search.

Input: Training set $\mathcal{T} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, beam width b
Output: Model parameters $\hat{\theta}$

$B^{(0)} = \{(1, 0)\}$ {initialize beam}
for $j = 1 \dots K$ **do**
  $B^{(j)} = \{\}$
  for (parentTags, parentScores) $\in B^{(j-1)}$ **do**
    for $i \in \{1, \dots, K\} \backslash$parentTags **do**
      $\mathcal{K} \leftarrow k\left(\phi(x, y_{[\text{parentTags}]}), \phi(x', y'_{[\text{parentTags}]})\right)$
      $\ell = [y_i^{(1)}; y_i^{(2)}; \dots; y_i^{(m)}]$
      if $\text{KTA}(\mathcal{K}, \ell) > \min\{v : (\cdot, v) \in B^{(j)}\}$ **then**
        $B^{(j)} \leftarrow B^{(j)} \cup (\text{parentTags} \cup \{i\}, \text{KTA}(\mathcal{K}, \ell))$
        $B^{(j)} \leftarrow \text{Top-}b(B^{(j)})$
      end if
    end for
  end for
end for
$\hat{v} = \text{argmax}_v\{v : (\cdot, v) \in B^{(K)}\}$ {highest score}
Return $\hat{\theta}$ learned by training a PCC using the ordering specified by $\hat{\pi} : (\hat{\pi}, \hat{v}) \in B^{(K)}$

$$\text{KTA}(\mathcal{K}, y) = \frac{\langle \mathcal{K}, yy^\top \rangle_F}{\sqrt{\langle \mathcal{K}, \mathcal{K} \rangle_F \langle yy^\top, yy^\top \rangle_F}} \,.$$

In practice, the KTA score may be much more efficient to compute than training a (partial) PCC. (Indeed, this was our experience in the empirical study reported in Section 5.) Note that there are also hidden costs with training a classifier, such as performing cross-validation to determine regularization and other hyperparameters. Intuitively, the KTA score can be considered as a proxy for the accuracy of a classifier trained on the same input features and outputs and therefore it is reasonable to expect the KTA scores to correlate positively with the accuracies of a classifier.

The KTA score of a vertex v at level t is computed by constructing a kernel matrix whose entries are $k\left(\phi(x, y_{\pi_v([t-1])}), \phi(x', y'_{\pi_v([t-1])})\right)$, where $k(\cdot, \cdot)$ is the kernel function, $\phi(x, z) = x \otimes z$, i.e., the Kronecker product of x and z, for *cross-product features* and $\phi(x, z) = x \oplus z$ for *concatenated features*, and $y_{\pi_v([t-1])} = (y_{\pi_v(1)}, \dots, y_{\pi_v(t-1)})$. For linear kernels, the kernel matrix factorizes into the product of the kernel matrix defined on the input features and the kernel matrix defined on the output labels. Note that earlier, the log-likelihood scoring function led to a naturally additive objective function. While the same can be done with KTA, it is an empirical question whether this will be appropriate or not. Observe in particular that an alternative is to use the product of KTA scores, which treats the KTA as a surrogate for the raw probability score itself.

Table 1. Details of benchmark multilabel data sets [16]

Data set	# training inst. (m)	# test inst.	# features (d)	# tags (K)
Emotions	391	202	72	6
Scene	1211	1196	294	6
Yeast	1500	917	103	14
Genbase	463	199	1186	27
Medical	333	645	1449	45
Enron	1123	579	1001	53

5 Experimental Results

5.1 Data Sets and Methods

We report experiments on benchmark multilabel data sets [16] listed in Table 1. We selected all the data sets with fewer than 100 tags and fewer than 10K training instances so that all models can be trained easily using batch optimization methods. Note that on the majority of these data sets, inference by exhaustive enumeration is either computationally expensive or intractable. All data sets have a pre-defined test set, and our reported results are on this set. We compare the following *reduction*-based algorithms:

(a) Binary relevance (BR): This is the baseline algorithm where we train separate independent logistic regressors for each tag.
(b) Kernel dependency estimation (KDE): This is the algorithm described in [4]. Here, a (linear) transformation using PCA is applied to the original label matrix in order to decorrelate the tags. Then, independent regressors are trained in the transformed label space.
(c) Probabilistic classifier chain (PCC): We use the original formulation as described in [1] but with beam search as inference and the original ordering of tags found in the data sets.
(d) PCC with logistic regression using beam search for *both* learning the tag ordering and inference. To learn the tag ordering, we use kernel target alignment as the scoring function in beam search. Note that, in this setting, the output of beam search for learning is the tag ordering which is then used at a later stage to train a PCC.

We evaluate the performance of algorithms using the following loss functions:

(i) **Subset** $0/1$ **loss:**

$$\ell_{0/1}(y, \hat{y}) = [\![y \neq \hat{y}]\!] \ ,$$

(ii) **Hamming loss:**

$$\ell_h(y, \hat{y}) = \sum_{i=1}^{K} [\![y_i \neq \hat{y}_i]\!] \ , \quad \text{and}$$

(iii) **Ranking loss:**

$$\ell_r(y, \hat{y}) = \sum_{(i,j):y_i>y_j} \left([\![\hat{y}_i < \hat{y}_j]\!] + \frac{1}{2}[\![\hat{y}_i = \hat{y}_j]\!] \right) ,$$

where y and $\hat{y}$ are the target and the predicted labels respectively. It has been noted previously that BR is a strong baseline on a number of loss functions [2,1]. (It is in fact theoretically optimal for Hamming and ranking losses [1].) We have found this to be especially true if the base classifier for each tag is regularized. Some previous studies, such as [1], use an unregularized base classifier, for which BR may be misleadingly sub-optimal. In all our experiments, we used *regularized* linear models and tuned the regularization parameter using cross-validation.

5.2 Results and Analysis

Q1: What is the effect of beam width used in inference and learning tag orderings on the performance of PCC?

For three of the data sets, namely, *Emotions*, *Scene* and *Yeast*, it is possible to do inference by exhaustive enumeration of all possible labels. We compared the performance of (i) binary relevance (BR) and (ii) probabilistic classifier chain (PCC) with the original tag ordering and using beam search for inference for several values of beam width b. Figure 3 shows the performance of the algorithms measured in terms of subset 0/1 loss, Hamming loss and ranking loss with varying beam width.

From the figures, we see that the test set performance of PCC converges rapidly with $b < 15$ to the performance obtained with exhaustive enumeration, especially for the subset 0/1 loss. We also observed that with $b = 15$, more than 90% of the correct labels were predicted within the candidate labels found by beam search, even if the single best label found by beam search was not exactly correct. Note that for Hamming and ranking losses, the loss at certain values of beam width is lower than the loss obtained by exhaustive enumeration which is surprising at first glance since exhaustive enumeration should ideally provide a lower bound for the test set loss. However, note that in beam search, labels are scored according to the joint probability $p(y \mid x)$ which may not necessarily correspond to the optimal result. Indeed, one of the points made in [1] was that taking $\text{argmax}_{y \in \mathcal{Y}} p(y \mid x)$ gives the optimal result for subset 0/1 loss, but for Hamming and ranking losses the optimal result is for tag k, i.e., $\text{argmax}_{b \in \{0,1\}} p(y_k = b \mid x)$, which could be different in the case of misspecified models.

We also analyzed the effect of increasing beam width used in beam search to determine the tag ordering on the classifier performance. Figure 4 shows the performance of PCC with varying beam width on the three data sets, *Emotions*, *Scene* and *Yeast*. The test time inference was done by exhaustive enumeration of all possible labels. From the figure, we see that there is no clear pattern with varying beam width for subset 0/1, Hamming and ranking losses. However, we note that negative log-likelihood is non-increasing with increasing beam width.

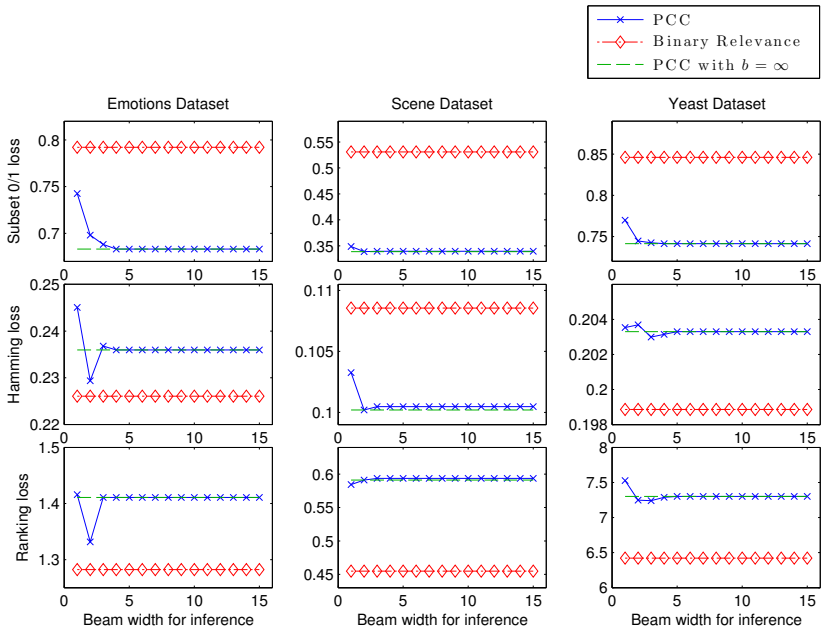


Fig. 3. Effect of beam width used in inference on the performance of classifiers

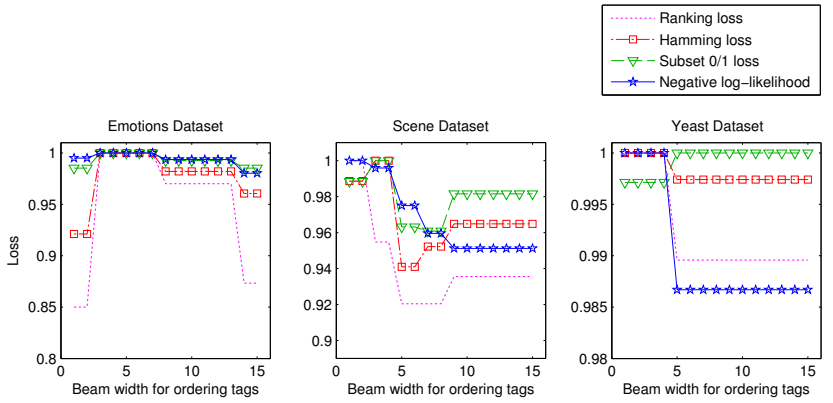


Fig. 4. Effect of beam width used to learn the tag ordering on the performance of PCCs. All losses have been scaled to $[0, 1]$ for the sake of legibility.

This confirms that beam search using KTA as the scoring function is successful at finding orderings that lead to good negative log-likelihood; as the amount of search done by beam search increases, better orderings are found.

Table 2. Test set performance of binary relevance (BR), kernel dependency estimation (KDE) and probabilistic classifier chain (PCC) trained with *cross-product features* measured w.r.t. subset 0/1 loss (*top*), Hamming loss (*middle*) and ranking loss (*bottom*) on the benchmark data sets. Numbers in subscript and superscript indicate beam width used to learn tag ordering and for inference respectively. PCC_{or} is PCC trained with original tag ordering. The last row in each table shows the ranking of algorithms averaged across all the data sets, with lower ranks being better.

	BR	KDE	PCC^1_{or}	PCC^5_{or}	PCC^{15}_{or}	PCC^1_1	PCC^5_1	PCC^5_5	PCC^{15}_{15}
Scene	0.5309	0.6204	0.3487	0.3395	0.3395	0.3829	0.3620	0.3495	0.3478
Yeast	0.8462	0.8397	0.7699	0.7416	0.7416	0.7666	0.7579	0.7590	0.7601
Emotions	0.7921	0.7822	0.7426	0.6832	0.6832	0.6832	0.6733	0.6733	0.6634
Enron	0.8774	0.9016	0.8273	0.8169	0.8169	0.8394	0.8048	0.8100	0.8100
Medical	0.4170	0.4310	0.3891	0.3643	0.3643	0.3798	0.3597	0.3597	0.3597
Genbase	0.0201	0.0201	0.0201	0.0201	0.0201	0.0201	0.0201	0.0201	0.0201
Avg. Rank	7.83	8	6	3.67	3.67	6	3.25	3.5	3.08

	BR	KDE	PCC^1_{or}	PCC^5_{or}	PCC^{15}_{or}	PCC^1_1	PCC^5_1	PCC^5_5	PCC^{15}_{15}
Scene	0.1086	0.1204	0.1033	0.1005	0.1005	0.1189	0.1105	0.1044	0.1058
Yeast	0.1989	0.1984	0.2035	0.2033	0.2035	0.2154	0.2106	0.2094	0.2098
Emotions	0.2261	0.2236	0.2450	0.2360	0.2360	0.2351	0.2302	0.2302	0.2211
Enron	0.0463	0.0465	0.0488	0.0506	0.0508	0.0507	0.0516	0.0507	0.0517
Medical	0.0122	0.0127	0.0148	0.0132	0.0132	0.0122	0.0114	0.0114	0.0114
Genbase	0.0010	0.0008	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
Avg. Rank	3.67	3.5	5.67	4.83	5.58	6.42	5.83	4.58	4.92

	BR	KDE	PCC^1_{or}	PCC^5_{or}	PCC^{15}_{or}	PCC^1_1	PCC^5_1	PCC^5_5	PCC^{15}_{15}
Scene	0.4548	0.5084	0.5844	0.5936	0.5936	0.6112	0.5920	0.5610	0.5284
Yeast	6.4209	6.4384	7.5267	7.3021	7.3010	7.6194	7.5463	7.4297	7.4526
Emotions	1.2822	1.4307	1.4158	1.4109	1.4109	1.5347	1.4851	1.4851	1.2970
Enron	12.9378	14.7219	16.1105	16.8929	16.9326	16.4447	16.9197	16.8765	16.6123
Medical	1.6271	1.3659	3.6922	3.7186	3.7202	2.9093	3.0783	3.0682	3.0682
Genbase	0.1709	0.0452	0.1910	0.1910	0.1910	0.1834	0.1834	0.1884	0.1884
Avg. Rank	1.33	2.33	5.83	6.33	6.67	6.25	6.5	5.42	4.33

Q2: Does learning to order tags improve the performance of PCC when compared to PCC using a random or otherwise pre-defined ordering?

The results are shown in Table 2. All variants of PCC consistently outperform or are in par with binary relevance for the subset 0/1 loss. Note that PCC^1_{or} is the variant of PCC which uses the original tag ordering in the data sets and greedy search for inference, i.e., beam search with $b = 1$. All variants of PCC that uses beam search for inference and/or beam search to determine the tag ordering outperform PCC^1_{or} which clearly demonstrates the advantages of using beam search for PCC. On a majority of the data sets, variants of PCC that used beam

search to determine the tag ordering using KTA as the scoring function gave the best results. For Hamming and ranking losses, we found that binary relevance is a strong baseline and outperformed PCC on average thus also confirming the results reported in [1]. Nevertheless, as for the subset 0/1 loss, we found that PCC using beam search to determine the tag ordering performed, on average, better than PCC that used the original tag ordering.

An interesting observation from the results reported in [1] is that for Hamming and ranking losses, PCC performs worse than binary relevance on average, but an ensemble of PCCs that were created from a random subsample of tag orderings performed better than binary relevance. As noted in [1], comparing a non-ensemble method with ensemble methods is not fair and we suspect that the improvements in performance from using an ensemble of PCCs may be due to

Table 3. Test set performance of binary relevance (BR), kernel dependency estimation (KDE) and probabilistic classifier chain (PCC) trained with *concatenated features* measured w.r.t. subset 0/1 loss (*top*), Hamming loss (*middle*) and ranking loss (*bottom*) on the benchmark data sets. The last row in each table shows the ranking of algorithms averaged across all the data sets, with lower ranks being better.

	BR	KDE	PCC^1_{or}	PCC^5_{or}	PCC^{15}_{or}	PCC^1_1	PCC^5_1	PCC^5_5	PCC^{15}_{15}
Scene	0.5309	0.6204	0.4022	0.3863	0.3863	0.4022	0.3813	0.3855	0.3813
Yeast	0.8462	0.8397	0.7895	0.7634	0.7634	0.8070	0.7612	0.7601	0.7601
Emotions	0.7921	0.7822	0.7475	0.6832	0.6832	0.7228	0.6634	0.6634	0.6634
Enron	0.8774	0.9016	0.8670	0.8497	0.8497	0.8566	0.8411	0.8411	0.8428
Medical	0.4170	0.4310	0.4093	0.4000	0.4000	0.4124	0.4031	0.4031	0.4031
Genbase	0.0201	0.0201	0.0201	0.0201	0.0201	0.0201	0.0201	0.0201	0.0201
Avg. Rank	7.83	8	6.25	4.08	4.08	6.25	2.83	2.83	2.83

	BR	KDE	PCC^1_{or}	PCC^5_{or}	PCC^{15}_{or}	PCC^1_1	PCC^5_1	PCC^5_5	PCC^{15}_{15}
Scene	0.1086	0.1204	0.1145	0.1113	0.1113	0.1104	0.1086	0.1095	0.1056
Yeast	0.1989	0.1984	0.2131	0.2092	0.2092	0.2204	0.2113	0.2106	0.2106
Emotions	0.2261	0.2236	0.2368	0.2261	0.2261	0.2228	0.2112	0.2129	0.2129
Enron	0.0463	0.0465	0.0465	0.0462	0.0462	0.0461	0.0461	0.0461	0.0461
Medical	0.0122	0.0127	0.0122	0.0120	0.0120	0.1216	0.1196	0.1196	0.1196
Genbase	0.0010	0.0008	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
Avg. Rank	4.58	4.92	7.08	4.92	4.92	5.83	4.25	4.5	4

	BR	KDE	PCC^1_{or}	PCC^5_{or}	PCC^{15}_{or}	PCC^1_1	PCC^5_1	PCC^5_5	PCC^{15}_{15}
Scene	0.4548	0.5084	0.6120	0.5978	0.5978	0.5397	0.5485	0.5293	0.5000
Yeast	6.4209	6.4384	7.7121	7.5071	7.5071	7.5474	7.6619	7.6314	7.6336
Emotions	1.2822	1.4307	1.3911	1.3218	1.3218	1.3317	1.3564	1.3366	1.3366
Enron	12.9378	14.7219	13.0743	13.0846	13.0846	13.0708	13.0656	13.0639	13.0639
Medical	1.6271	1.3659	1.7798	1.7752	1.7752	1.7395	1.7333	1.7333	1.7333
Genbase	0.1709	0.0452	0.1859	0.1859	0.1859	0.1884	0.1884	0.1884	0.1884
Avg. Rank	1.33	4.17	7.5	5.42	5.42	5.42	6.08	4.92	4.75

the *ensemble* effect. Due to these reasons, we do not report results for ensembles of our method. The goal in this paper is to conduct a fair, controlled comparison of our beam search approach to the original PCC approach. Note that it is possible to create an ensemble of PCCs in our approach by selecting a subset of best scoring leaf vertices from the beam search tree used to determine the tag ordering.

Q3: How much of an impact does cross-product features have on the performance of PCC when compared to concatenated features?

The results are shown in Table 3. The relative performance of different algorithms is similar to those reported in Table 2. For the subset 0/1 loss, we found improvements in performance when using cross-product features, $\phi(x, y) = x \otimes y$. However, for Hamming and ranking losses, cross-product features seem to degrade the performance of classifiers when compared to features formed by concatenating labels, $\phi(x, y) = x \oplus y$. The fraction of experiments (an experiment is an entry in Table 2 or 3) where cross-product features performed better than concatenated features were 0.82, 0.44 and 0.31 (with ties broken at random) for subset 0/1, Hamming and ranking loss respectively. We suspect this rather surprising negative result for Hamming and ranking losses may be due to overfitting with increased number of features in the cross-product feature representation. However, the interplay between choice of loss functions and the choice of feature representations is unclear and we leave this as an open question.

6 Concluding Remarks

Empirical results clearly demonstrate the benefit of using beam search for test time inference and to learn the tag ordering. We believe these are important extensions to probabilistic classifier chains. Regarding directions for future work, one general issue with multilabel classification is that most data sets are highly imbalanced at the tag level, i.e., every tag has very few positive instances. Using logistic regression gives biased probability estimates on imbalanced datasets [17]. Since probabilistic classifier chains rely on predicting accurate probabilities for each classifier in the chain, such biased estimates may hamper overall performance. At a minimum, we believe better results can be obtained by post-processing the scores by isotonic regression [18,19].

Another issue is concerned with using cross-product features for large number of tags where learning linear models may pose scalability issues. To circumvent this problem, we may compute the cross-product (linear) kernel matrix, and, if the number of training instances is not high, use a kernel method (kernel SVM, kernel logistic regression). Otherwise, we can compute a low-dimensional representation of the feature space given the kernel matrix using, for example, kernel PCA. An alternative (approximation) is to treat the rows (or columns) of the kernel matrix as features – the so-called empirical kernel map [20] – and train a linear SVM or a linear logistic regression using these features.

References

1. Dembczyński, K., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: ICML (2010)
2. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning 85(3), 333–359 (2011)
3. Finley, T., Joachims, T.: Training structural SVMs when exact inference is intractable. In: ICML (2008)
4. Weston, J., Chapelle, O., Elisseeff, A., Schölkopf, B., Vapnik, V.: Kernel dependency estimation. In: NIPS (2002)
5. Rai, P., Daumé III, H.: Multi-label prediction via sparse infinite CCA. In: NIPS (2009)
6. Hsu, D., Kakade, S., Langford, J., Zhang, T.: Multi-label prediction via compressed sensing. In: NIPS (2009)
7. Bi, W., Kwok, J.T.: Multilabel classification on tree- and DAG-structured hierarchies. In: ICML (2011)
8. Cristianini, N., Shawe-Taylor, J., Elisseeff, A., Kandola, J.S.: On kernel-target alignment. In: NIPS (2001)
9. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining 3(3), 1–13 (2007)
10. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer (2010)
11. Sorower, M.S.: A literature survey on algorithms for multi-label learning. Technical report, Oregon State University, Corvallis, OR, USA (December 2010)
12. Dembczyński, K., Waegeman, W., Hüllermeier, E.: Joint mode estimation in multilabel classification by chaining. In: ECML Workshop - CoLISD (2011)
13. Zaragoza, J., Sucar, L., Morales, E.: Bayesian chain classifiers for multidimensional classification. In: IJCAI (2011)
14. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice-Hall, Englewood Cliffs (2003)
15. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)
16. Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Vlahavas, I.: Mulan: A java library for multi-label learning. Journal of Machine Learning Research 12, 2411–2414 (2011)
17. King, G., Zeng, L.: Logistic regression in rare events data. Political Analysis 9(2), 137–163 (2001)
18. Zadrozny, B., Elkan, C.: Transforming classifier scores into accurate multiclass probability estimates. In: KDD (2002)
19. Menon, A.K., Jiang, X., Vembu, S., Elkan, C., Ohno-Machado, L.: Predicting accurate probabilities with a ranking loss. In: ICML (2012)
20. 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)

Learning Multiple Tasks with Boosted Decision Trees

Jean Baptiste Faddoul[2], Boris Chidlovskii[1], Rémi Gilleron[2], and Fabien Torre[2]

[1] Xerox Research Center Europe
boris.chidlovskii@xrce.xerox.com
http://www.xrce.xerox.com
[2] Lille University, LIFL and INRIA Lille Nord Europe
{jean-baptiste.faddoul,fabien.torre,remi.gilleron}@univ-lille3.fr
http://www.lifl.fr

Abstract. We address the problem of multi-task learning with no label correspondence among tasks. Learning multiple related tasks simultaneously, by exploiting their shared knowledge can improve the predictive performance on every task. We develop the multi-task Adaboost environment with Multi-Task Decision Trees as weak classifiers. We first adapt the well known decision tree learning to the multi-task setting. We revise the information gain rule for learning decision trees in the multi-task setting. We use this feature to develop a novel criterion for learning Multi-Task Decision Trees. The criterion guides the tree construction by learning the decision rules from data of different tasks, and representing different degrees of task relatedness. We then modify MT-Adaboost to combine Multi-task Decision Trees as weak learners. We experimentally validate the advantage of the new technique; we report results of experiments conducted on several multi-task datasets, including the Enron email set and Spam Filtering collection.

Keywords: Multi-Task Learning, Boosting, decision trees, information gain.

1 Introduction

Multi-task learning [3] aims at improving the performance of related tasks by learning a model representing the common knowledge across the tasks. Traditionally, the existing techniques assume that tasks share the same instance and label space [14], in the case of classification [6,18], regression [5,1], ranking [4] and feature learning [2].

However, in many natural settings these assumptions are not satisfied. A known example is the automatic categorization of Web pages into hierarchical directories, like DMOZ or Yahoo! [12]. When building a categorizer for the Yahoo! directory, it is desirable to take into account DMOZ web directory, and vice versa. The two tasks are clearly related, but their label sets are not identical. Moreover, both ontologies can evolve with time when new categories are added to the directories and some old categories die naturally due to lack of interest.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 681–696, 2012.

Multi-task learning with no label correspondence was considered in Quadrianto et al. in [15], where the problem is formulated as learning the maximum entropy estimator $H(Y|X)$ for each task while maximizing the mutual information $-H(Y, Y')$ among the label sets Y and Y' of different tasks. Their approach relies on the hypothesis of the global correlation between tasks in the whole learning space. Tests on the real world datasets show however that this global relatedness assumption turns to be too strong. Indeed, task relatedness may show up different degrees or even different signs in different regions of the learning space. It is therefore important that the multi-task learner copes with the varying relatedness of tasks, learns its different degrees and accommodates the inductive bias accordingly.

We are interested in the multi-task learning where label sets are close but differ from one task to another and the number of classes might be different across tasks. Our motivating example is the automatic classification of e-mails in personal inboxes [13]. Similarly to the case of Yahoo! and DMOZ web directories, categories used in two e-mail inboxes may be related but not identical. For example, people may use *Family* or *Home* categories for personal e-mails and *Finance* or *Budget* for e-mails relevant to financial issues. The application becomes particularly critical when inboxes are owned by people from the same organization; they may share the same messages but classify them according to personal category names. We therefore expect that learning all tasks simultaneously can benefit to the classification model for each task.

In the previous work [7] we proposed a method for multi-task learning for tasks with different label sets which makes no assumption on global relatedness. For this purpose, we developed a multi-task learning algorithm *(MT-Adaboost)* which extends Adaptive boosting *(Adaboost)* [9] to the multi-task setting. The boosting technique is used to generate and combine multiple (weak) classifiers to improve the predictive accuracy. As weak classifiers, we introduced multi-task stumps which are trees having at each node a decision stump for one task. According to the boosting principle, a smart re-weighting of examples from different tasks without label correspondences can grasp the local relatedness of tasks. The method however suffers from some limitations. The greedy algorithm which learns a multi-task stump level-by-level, is based on a heuristic choosing at the root the best stump for the easiest task (where the training error is the lowest); then it forwards recursively to the next levels to learn the remaining tasks. In this kind of a cascade classification on tasks, it learns at each node a classifier for a task taking into account the other tasks' classifiers in the node's ancestors. Unfortunately, such a sequential design of multi-task stumps performs poorly when its greedy algorithm fails to capture task relatedness. In addition, multi-task stumps are binary classifiers, and their extension to multiple multi-class tasks requires additional efforts.

In this work, we propose a novel technique for the multi-task learning which addresses the limitations of previous approaches. First, we propose *Multi-Task Decision Tree (`MT-DT`)* as a multi-task weak classifier. We revisit the well known *C4.5* decision tree learning and adapt it to the multi-task setting. Decision trees

are naturally multi-class classifiers, thus MT-DT can learn multiple multi-class classification tasks. Our main contribution is in proving that MT-DT can benefit from an improved information gain criterion due to the multi-tasking. Unlike multi-task stumps, the criterion used to learn the nodes makes use of the data from several tasks at each node.

Second, we proceed by plugging the MT-DT in the boosting framework; we modify MT-Adaboost to cope with the multi-class problems accordingly. We follow the work of Schapire et al. [17] which analyzed Adaboost with weak learners which abstain and proposed several variations of Adaboost to carry on multi-class problems. Our modification of MT-Adaboost adapts their *Adaboost.M1* algorithm.

As the experimental study will show, our method does not have a uniform margin of improvement over all the tasks. In other words, the tasks which have lower prediction accuracy in single task learning they have a higher improvement potential when learned by our multi-task algorithm.

In the following section we formalize the multi-task learning and introduce the multi-task decision trees. We revisit the information gain rule for the multi-task learning and derive a novel criterion for learning MT-DT. In Section 3 we present the boosting framework for the multi-task with MT-DT as weak learners. We report the evaluation results for synthetic and real world multi-task datasets in Section 4; Section 5 concludes the paper.

2 Multi-Task Learning

2.1 Notation and Setting

In the conventional setting, a supervised classification task T is defined over the instance space $\mathcal{X}$ and the space $\mathcal{Y}$ of labels. Let D denote a distribution over $(\mathcal{X}, \mathcal{Y})$, let $f : \mathcal{X} \to \mathcal{Y}$ be a target function. Given a set of training examples $S = \{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i = f(x_i),\ 1 \leq i \leq m\}$, the goal of learning is to find an hypothesis function h which minimizes an error function, defined over D as $\text{error}(h) = Pr_{<x,y>\sim D}[h(x) \neq y]$.

We now consider N classification tasks $T_1, \ldots, T_N$ over the instance space $\mathcal{X}$ and label sets $Y_1, \ldots, Y_N$, where labels in sets $\mathcal{Y}_i$ are correlated but not identical. Due to the label mismatch the label sets, we assume that $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$. We are interested in solving N classification tasks simultaneously, in order to improve classification accuracy. We suppose a distribution D^N over $\mathcal{X} \times \{1, \ldots, N\}$. We assume that, for every $j \in \{1, \ldots, N\}$, the projection on the distribution's j-th component will correspond to the original distribution for task T_j. A multi-task classification algorithm will take as input the training set $S = \{< x_i, y_i, j > \mid x_i \in \mathcal{X},\ y_i = f_j(x_i) \in \mathcal{Y}_j,\ j \in \{1, \ldots, N\},\ 1 \leq i \leq m\}$. It should be noted that the same instance x can appear in sample S in different tasks T_i and T_j with corresponding labels y_i and y_j. The goal is to find an hypothesis $h : \mathcal{X} \to \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_N$ which minimizes $\text{error}(h) = Pr_{<x,y,j>\sim D^N}[h_j(x) \neq y]$, where $h_j(x)$ is the j-th component of $h(x)$ and $j \in \{1, \ldots, N\}$.

2.2 Multi-Task Decision Tree

Decision tree learning is a well known technique in machine learning; it uses a decision tree as a predictive model which maps observations from the instance space to the target values. In the case of classification, tree leaves represent class labels and branches represent conjunctions of item attributes that lead to those class labels [16].

In the C4.5 and C5.0 tree generation algorithms, the decision tree learning uses the concept of the *information gain* (IG) from the information theory. At the root of the tree, the algorithm chooses an attribute that yields the highest IG on the training set. Such an attribute splits the training set S into two subsets S_1 and S_2 whose sum of labels entropy is the lowest. The algorithm then recursively applies the information gain rule on the subsets. The recursion is stopped when all items of a subset have the same label, a decision leaf corresponding to this label [1].

The amount of information gain about a label variable $Y \in \mathcal{Y}$ obtained by observing that an attribute variable a takes value v can be measured by the Kullback-Leibler divergence $D_{KL}(p(Y|a)||p(Y))$ of the prior distribution $p(Y)$ from the posterior distribution $p(Y|a)$ for Y given a. The information gain rule estimates the average improvement. Thus the decision tree algorithm uses the rule to recursively split the instance space, by selecting an attribute with the high information gain.

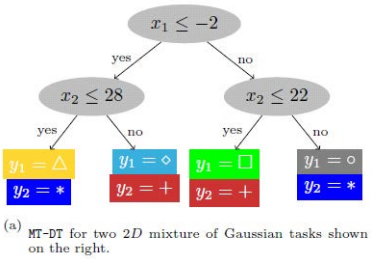
In this paper we adapt the information gain based decision tree learning to the multi-task setting. One obvious difference between one- and multi-task setting is in the tree structure. One-task decision tree uses the internal test nodes to guide the decision process while the final decision on assigning a label to a sample is made in a tree leaf.

The structure of an multi-task decision tree (`MT-DT`) is different in the way it guides the decision process for multiple tasks. This process is not necessarily the same for all tasks. An `MT-DT` can make a final decision for some tasks in an internal test node, not a tree leaf. This happens when the internal test node has enough information to classify an instance of a certain task T, in such a case a decision leaf with the appropriate classification decision for T is added to the tree and the learning proceeds with the remaining tasks.

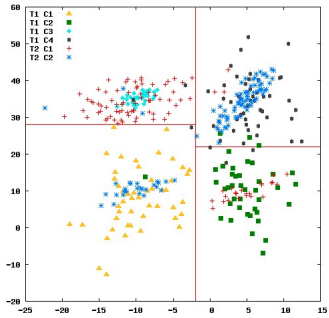
Figure 1.a gives an example of an `MT-DT` learned for two synthetic tasks generated from `2D` mixture of Gaussians (see Figure 1.b). T_1 has four labels ($\mathcal{Y}_1 = \{\square, \diamond, \triangle, \circ\}$) and T_2 has two labels ($\mathcal{Y}_2 = \{+, *\}$). Two labels of T_1 ($\square, \diamond$) are correlated with label $+$ of T_2, while two other labels of T_1 ($\triangle, \circ$) are correlated with label $*$ of T_1. The generated `MT-DT` has three internal test nodes and each decision leaf carries one rule per task.

Another example of `MT-DT` is showed in Figure 2. Task T_1 is the same as Figure 1, while task T_2 is generated differently from a mixture of Gaussians (see Figure 2.b). This results in a different correlation pattern between the tasks. The learned `MT-DT` has an early decision leaf for T_2 since knowing that $x_1 > -2$ is enough to predict the label class $*$ for T_2.

[1] Some pruning is often used to generalize the rules learned to unobserved items.

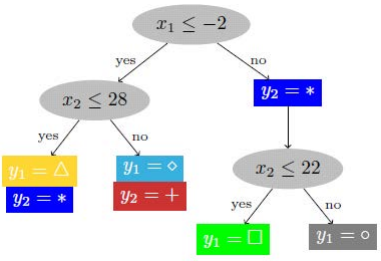


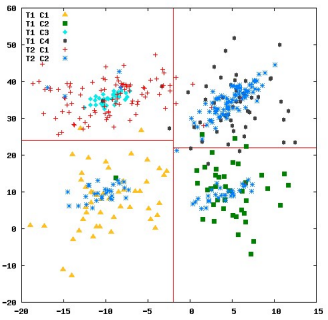
(a) MT-DT for two $2D$ mixture of Gaussian tasks shown on the right.



b) a)

Fig. 1. a) MT-DT example for two tasks. b) Two $2D$ mixture of Gaussian tasks.





b) a)

Fig. 2. a) MT-DT with early decision leaf. b) Two $2D$ mixture of Gaussian tasks.

When moving from one- to multi-task setting, the adaptation of the tree structure learning is straightforward. The main challenge is however in the optimal way of using the information gain criteria. In the next section we show how MT-DT can profit from the multi-task setting.

2.3 Multi-Task Information Gain

As said before the decision tree learning is based on the entropy-based criteria, in particular, on the quantity of the mutual dependency between two random variables, the label variable $Y \in \mathcal{Y}$ and the observation attribute a which is one of the attributes of an input vector $x \in \mathcal{X}$. The information gain denoted $IG(Y;a)$ can be expressed as follows

$$IG(Y;a) = H(Y) - H(Y|a), \tag{1}$$

where $H(Y) = -\sum_{y \in \mathcal{Y}} p(y) \log p(y)$ is the marginal entropy of label set $\mathcal{Y}$ and $H(Y|a) = \sum_v p(v) H(Y|a = v)$ is the conditional entropy of Y knowing a.

Assume now we cope with N tasks with the corresponding label sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_N$, respectively. For learning the `MT-DT`, the baseline approach is to treat all the tasks together by concatenating the label sets, denoted as $\oplus_{j=1}^{N} \mathcal{Y}_j$. The concatenated task takes as input a sample $S = \{< x_i, y_i > | \ x_i \in \mathcal{X}, \ y_i = f(x_i) \in \oplus_{j=1}^{N} \mathcal{Y}_j, \ 1 \leq i \leq m\}$. It can use the *joint information gain* for learning decision rules, defined as $IG_J = IG(\oplus_{j=1}^{N} Y_j; a)$. As an alternative to IG_J, we could use the unweighted sum of individual task information gains, $IG_U = \sum_{j=1}^{T} IG(Y_j; a)$. Evaluations however show that IG_U gives lower information gain values comparing to IG_J.

We will prove below that IG_J is equivalent to the weighted sum of individual task information gains and infer an IG criterion with higher values compared to IG_J. The novel IG criterion, denoted IG_M, takes the maximum value among the individual IGs, $IG_M = max\{IG(Y_j; a), j = 1, \ldots, N\}$.

We first recall the generalized grouping feature of the entropy [10] in the following lemma. It establishes a relationship between the entropy of an entire set of values and the entropies of its disjoint subsets.

Lemma 1. *For $q_{kj} \geq 0$, such that $\sum_{k=1}^{n} \sum_{j=1}^{m} q_{kj} = 1, p_k = \sum_{j=1}^{m} q_{kj}, \forall k = 1, \ldots, n$, the following holds*

$$H(q_{11}, \ldots, q_{1m}, q_{21}, \ldots, q_{2m}, \ldots, q_{n1}, \ldots, q_{nm}) = \tag{2}$$

$$H(p_1, \ldots, p_n) + \sum p_k H\left(\frac{q_{k1}}{p_k}, \ldots, \frac{q_{km}}{p_k}\right), p_k > 0, \forall k. \tag{3}$$

Using Lemma 1, we can prove the following theorem on the relationship between the joint information gain $IG(\oplus_{j=1}^{N} Y_j; a)$ of the full task set and of the individual tasks $IG(Y_j; a), j = 1, \ldots, N$.

Theorem 1. *For N tasks with the class sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_N$, let p_j denote the fraction of task j in the full dataset, $p_j = \frac{|S_j|}{\sum_{j=1}^{N} |S_j|}, j = 1, \ldots, N$, $\sum_{j=1}^{N} p_j = 1$. Then we have*

$$IG(\oplus_{j=1}^{N} Y_j; a) = \sum_{j=1}^{N} p_j IG(Y_j; a) \leq max(IG(Y_1; a), \ldots, IG(Y_N; a)). \tag{4}$$

Proof. First, we use Lemma 1 to develop the entropy term $H(\oplus_{j=1}^{N} Y_j)$ of the information gain (1). We have

$$H(\oplus_{j=1}^{N} Y_j) = H(p_1, \ldots, p_N) + \sum_{j=1}^{N} p_j H(Y_j), \tag{5}$$

where $\sum_{j=1}^{N} p_j = 1$.

Second, we develop the conditional entropy term in (1), as follows

$$H(\oplus_{j=1}^{N} Y_j|X) = \sum_x p(x) H(\oplus_{j=1}^{N} Y_j|a = v) \tag{6}$$

$$= \sum_v p(v) \left(H(p_1, \ldots, p_N) + \sum_{j=1}^{N} p_j H(Y_j|a = v) \right) \tag{7}$$

$$= H(p_1, \ldots, p_N) + \sum_{j=1}^{N} p_j H(Y_j|a). \tag{8}$$

Now we combine the entropy (5) and the conditional entropy (9) terms to evaluate the joint information gain $IG(\oplus_{j=1}^{N} Y_j; a)$. We obtain

$$IG(\oplus_{j=1}^{N} Y_j; a) = H(\oplus_{j=1}^{N} Y_j) - H(\oplus_{j=1}^{N} Y_j|a) \tag{9}$$

$$= \sum_{j=1}^{N} p_j IG(Y_j; a) \tag{10}$$

$$\leq max(IG(Y_1; a), \ldots, IG(Y_N; a)). \tag{11}$$

This completes the proof of the theorem.

Theorem 1 says that criterion IG_M for the decision tree learning in the multi-task case gives larger information gain values comparing to the joint one IG_J.

Figure 3 compares three criteria IG_U, IG_J and IG_M for some randomly generated two-task datasets. Two label sets are generated by sampling from the Uniform, Normal (with $\mu = 0$, $\sigma = 1$) and Poisson ($\lambda = 1$) distributions; the number of labels in the two sets vary from 2 to 20. Attributes values are sampled from uniform distributions in all cases. We measure the relative values of IG_M and IG_U with respect to IG_J. In all cases, we report the median, the upper and lower percentiles, and the whiskers over 100 runs. As the figure shows, IG_M yields on average up to 42% larger information gain values than IG_J, with the minimal gain in the case of two Uniform distributions.

As we can notice from the plots, the variance of IG_M values is very high compared to this of IG_U which have almost zero variance. An information gain criterion with small variance is not a good indicator to help choosing a good node because all node will have very close IG values. The explanation of such small variance is that when we take the sum of IGs for all tasks, it is difficult to come out with a node that satisfies all of them.

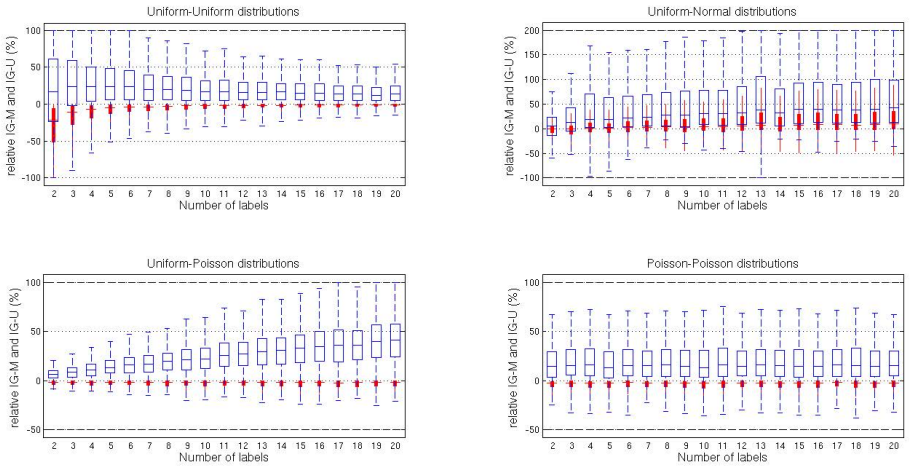


Fig. 3. Information gain for synthetic two-task datasets. The relative values of IG_M (in blue) and IG_U (in red).

2.4 Learning Algorithm for `MT-DT`

The learning algorithm for `MT-DT` applies one of proposed information gain criteria to the available training set S:

$$MTIG(S) \equiv (\mathbf{a}^*, v^*) = max_{a_{\in \mathcal{X}}, v \in V_a} IG_*(S),$$

where a is an attribute in feature space $\mathcal{X}$, S is the training set, a takes one of the possible values $v \in V_a$ and a pair (a^*,v^*) yields the optimal split on S using as criterion IG_* which can refer to IG_J, IG_U or IG_M.

The pseudo code of the MT-DT algorithm is presented in Algorithm 1. The algorithm makes a call to a function $MTIG$ which returns the node with rule $a \leq v$ that maximizes a given information gain on a multi-task training set S, Then it gets subsets S_1, S_2 resulting from splitting S on the chosen node. At each node the algorithm adds decision leaves for the tasks having no items in the subset or having items with the same label. Then, it calls recursively the procedure on each of subsets. The learning algorithm returns at the end a tree that gives for each example (x) one label for each task.

In the evaluation section, we test three versions of the IG criterion introduced before, IG_J, IG_U and IG_M. It is worth noting that we can limit the depth of the trees by modifying the stopping criterion, instead of stopping the growth of a certain branch when we have homogenous labels for all tasks in the subspace corresponding to that branch, we can stop when we exceed a threshold. For instance, when 80% of the examples are from the same labels. This should not be an issue as long as we are using an ensemble of trees learned by a boosting algorithm.

Require: $S = \cup_{j=1}^{N}\{e_i =< x_i, y_i, j >| \; x_i \in \mathcal{X};\; y_i \in \mathcal{Y}_j\}$
Require: $MTIG$: multi-task information gain criterion
`res = []` {Will contain the chosen node and early decision leaves, if any}
for $j = 1$ to N **do**
  if task j's examples (S_j) has all the same label **or** $S_j = \emptyset$ **then**
    Add to `res` a leaf for task j and label y. {y is either the unique label of S_j in case it is homogeneous or it is the majority label of its parent subset in case $S_j = \emptyset$}
    $S = S \setminus S_j$
  end if
end for
Get the `bestnode` rule $(a, v) = MTIG(S)$ which maximizes the information gain
Call `split`(S, a, v))
Get back $[S_1, S_2]$, two subsets resulted from splitting S based on `bestnode`
Add `bestnode` to `res`
Call recursively the algorithm on S_1 and S_2 to get the children of `res`
return `res`

Algorithm 1. `MT-DT` algorithm

3 Multi-Task Adaboost

In the previous section we developed a novel technique for learning `MT-DT`'s with an improved information gain criterion. To avoid all disadvantages of the decision trees such as overfitting, in this section we proceed by plugging the `MT-DT`'s in the boosting framework.

We adapt Adaboost.M1 which was introduced in [9]. We preferred M1 to MH or other multi-class boosting algorithm because it requires a weak classifier which is naturally multi-class. It does not need a weak learner which transforms a problem to several binary problems, and since, we propose a multi-task learner based on decision trees that are naturally multi-class classifiers, Adaboost.M1 is a good candidate. In addition to that, it is the most straightforward multi-class extention of Adaboost. Nevertheless, it puts strong requirement on the weak learner; actually, it requires the classification error of the weak classifier to be less than 0.5 w.r.t. to the current weight distribution, regardless the number of class labels. Some weak learners, such as stumps, are unable to satisfy such a strong boosting condition. But, normally, decision trees can satisfy this condition.

The proposed Multi-Task Adaboost algorithm (`MT-Adaboost`) is presented in Algorithm 2. T is the number of boosting iterations; `init` is a procedure to initialize the distribution D_1 over S; and `WL` is a weak learner that returns an `MT-DT` given as input a sample S and a distribution D over S. The final output is a multi-task classifier H from $\mathcal{X}$ into $\mathcal{Y}_1 \times \ldots \times \mathcal{Y}_N$. As in single task boosting algorithms, `MT-Adaboost` calls `WL` repeatedly in a series of rounds.

On each round t, the algorithm provides `WL` with the current distribution D_t and the training sample S, in return `WL` learns a classifier $h_t : \mathcal{X} \rightarrow \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_N$ which minimizes the training error on S with respect to D_t. The distribution D_{t+1} is then calculated from D_t and h_t as follows. Correctly classified examples by h_t will have their weights multiplied by $0 \leq \beta_t \leq 1$ (i.e., decreased), and the weights of misclassified examples will be left unchanged. Finally, the weights are renormalized by using the normalization constant Z_t.

The final classifier H for a given task j is a weighted vote of the weak classifiers' predictions for this task. The weight given to hypothesis h_t is defined to be $ln(1/\beta_t)$ so that greater weight is given to hypotheses with lower error. `MT-Adaboost` has the same theoretical properties of Adaboost.M1, that is, if the weak hypotheses have error only slightly better than 1/2, then the (training) error of the final hypothesis H drops to zero exponentially fast in function to the number of boosting iterations T.

4 Experiments

In this section we present a series of experiments on three datasets. We describe the datasets and evaluation framework, then we compare the predictive performance of single task decision trees to different MT-DTs learned using IG_J, IG_U and IG_M criteria. Then we report experimental results on boosted trees using MT-Adaboost.

Require: $S = \cup_{j=1}^{N} \{e_i =< x_i, y_i, j > | \; x_i \in \mathcal{X};\; y_i \in \mathcal{Y}_j\}$
1: $D_1 = \texttt{init}(S)$ *initialize distribution*
2: **for** $t = 1$ to T **do**
3: $h^t = \texttt{WL}(S, D_t)$ {*train the weak learner and get an hypothesis* `MT-DT` }
4: Calculate the error of h^t: $\epsilon_t = \sum_{j=1}^{N} \sum_{i:h_j^t(x_i) \neq y_i} D_j(x_i)$.
5: **if** $\epsilon_t > 1/2$ **then**
6: Set $T = t - 1$ and abort loop.
7: **end if**
8: $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
{*Update distribution:*}
9: **if** $h_j^t(x_i) == y_i$ **then**
10: $D_{t+1}(e_i) = \frac{D_t(e_i) \times \beta_t}{Z_t}$
11: **else**
12: $D_{t+1}(e_i) = \frac{D_t(e_i)}{Z_t}$
13: **end if**
14: **end for**
{*Where* Z_t *is a normalization constant chosen so that* D_{t+1} *is a distribution*}
15: **return** Classifier H defined by:

$$H_j(x) = \arg\max_{y \in \mathcal{Y}_j} (\sum_{i=1}^{i=T} (\ln 1/\beta_t)),\ 1 \leq j \leq N$$

Algorithm 2. MT-Adaboost

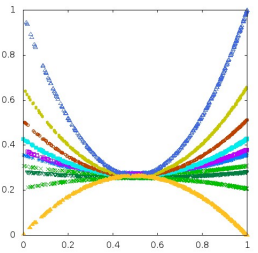
4.1 Datasets

Synthetic. We generate synthetically tasks with local relatedness patterns, by following the data generation technique described in [8]. Each pattern is generated a random Bayesian network (BN) from which one can derive different but related probabilistic distributions. The BN is created by generating (a) a random (directed acyclic) graph, (b) a set of functions (with random parameters) characterizing the dependence of every node on each one of its parents in the graph, and (c) a set of functions (with randomly assigned parameters) defining the probability density of each node [2].

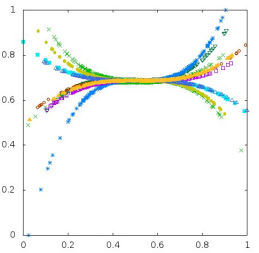
Figure 4 shows some examples of the local tasks relatedness generated using such method. In the plotted examples, the distributions feature cubic, exponential and linear correlation functions, with Beta, Gaussian and Laplacian densities. Using the random relatedness generator we generate three multi-task learning datasets. DS_1 consists of two tasks T_1 and T_2, having three and two labels, respectively. They are plotted in Figure 5.a. We can see that the red class of T_1 is locally correlated with the light blue class of T_2; similarly, the green class is locally correlated with the violet. However the dark blue class of T_1 which is locally correlated with the violet in the upper part of its density and with the light blue in the lower part. The second dataset DS_2 is shown in Figure reffig:mtsynthetic.b with tasks being also locally correlated. Finally, random noise is added to the labels of all tasks as follows. For a certain example with label y we place a discrete probability distribution over the label set with 90% of mass concentrated over y and the rest distributed equally over the other labels. Then we sample the noisy label from this distribution. It should be noted that we generate tasks with different number of class labels on purpose, in order to test the proposed methods on configurations not addressed by prior-art methods.

Enron. Enron dataset[3] contains all e-mails sent and received by some 150 accounts of the top management of Enron company and spans a period of several years. Annotations of the Enron dataset come from two different sources, thus, naturally constituting two tasks. The first is from the Department Of Justice of the United States DOJ[4], which has published a list of responsive emails used in the trials against the two CEO's of Enron. This set along with a manually annotated non-responsive emails constitute a binary classification task, *Responsive* Vs. *Non-responsive*, with total of 372 emails. The second annotated set comes from students of Berkeley University. Emails in this set are annotated by topic, for an average of 250 emails per topic. Five topics are used in our experiments: *Business*, *Legal*, *Influence*, *Arrangement* and *Personal*. We used the textual features of Enron dataset along with the social features (see [11] for more details).

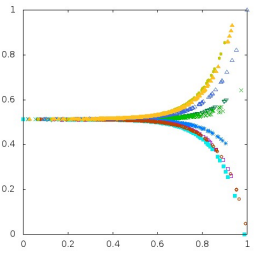
[2] The code is provided by Antonino Freno
http://researchers.lille.inria.fr/~freno/software.html
[3] http://www.cs.cmu.edu/~enron/
[4] http://www.usdoj.gov/enron/



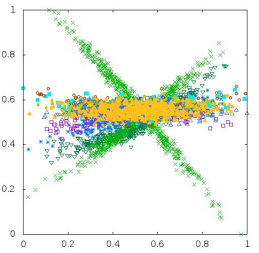
(a) A correlation pattern from beta-quadratic distributions



(b) A correlation pattern from gaussian-cubic distributions

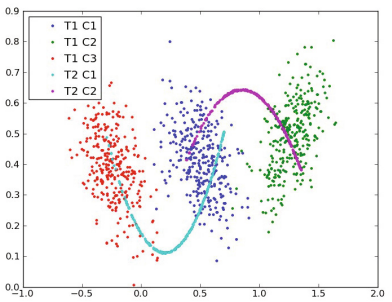


(c) A correlation pattern from gaussian-exponential distributions

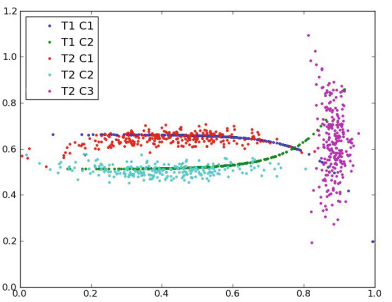


(d) A correlation pattern from gaussian-quadratic distributions

Fig. 4. Tasks Relatedness Patterns for synthetic 2D data



(a) Two related multi-class tasks



(b) Two related multi-class tasks

Fig. 5. Two classification problems, each with two multi-class tasks

Spam Filtering. This dataset was used for the ECML/PKDD 2006 discovery challenge. It contains email inboxes of 15 users. Each inbox has 400 spam/ham emails. They are encoded by standard bag-of-word vector representation. We consider each user as a task.

MNIST Character Recognition. We use this dataset adapted to the multi-task setting because it was used by a state-of-the-art method [15], so we can and we can compare with their results. For the experiments, we consider multi-task learning problems with 10 tasks representing the digits $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$. We follow the same protocol given in [15] so we can be able to have a good comparison.

4.2 Results on Trees

In this section we report experimental results of MT-DTs learned using IG_J, IG_U and IG_M criteria introduced in Section 2.3. We also compare MT-DT to single task decision trees learned with C4.5 algorithm. In all experiments we use the 5-fold cross validation, where each run consists of training on four folds and testing on the remaining one. We run all methods fifty times on random shuffles of the data and report the average values. Results in bold are statistically significant by a t-test with $\alpha = 0.05$.

Table 1 reports the size of training/test sets and the evaluation results for three synthetic datasets. We note that MT-DT with IG_M brings a significant improvement over C4.5. While IG_J and IG_U behave comparably to C4.5, they are slightly better on Task-1, but suffer an accuracy drop on Task-2.

Table 1. Average classification accuracy on the three synthetic datasets

Tasks	Train (Test)	C4.5	IG_J	IG_U	IG_M
Data Set 1					
Task-1	300 (1200)	86.432 ± 0.003	86.116 ± 0.063	86.070 ± 0.029	87.180 ± 0.037
Task-2	200 (1300)	89.532 ± 0.167	88.980 ± 0.391	89.237 ± 0.445	89.246 ± 0.341
Avg		87.982	87.548	87.653	**88.213**
Data Set 2					
Task-1	200 (1300)	90.738 ± 0.092	88.008 ± 0.606	89.848 ± 0.063	90.751 ± 0.085
Task-2	300 (1200)	83.525 ± 0.600	88.056 ± 1.363	88.221 ± 1.316	88.366 ± 0.366
Avg		87.132	88.032	89.035	**89.559**

The same behavior is observed on ECML'06 data (see Table 2). It shows a superiority of IG_M over other MT-DT criteria in accuracy values. However, learning tasks simultaneously does not bring the same improvement to all tasks, some tasks tend to benefit more from multi-task learning than others. Results show that more difficult tasks (tasks with a lower accuracy) got a higher improvement on their prediction accuracy when learned by other tasks.

Table 2. Average classification accuracy on three ECML'06 user inboxes

Tasks	Train (Test)	C4.5	IG_J	IG_U	IG_M
User-1	320 (80)	86.45 ± 1.23	86.19 ± 1.14	86.00 ± 1.88	**87.65±3.42**
User-2	320 (80)	85.13 ± 2.16	85.53 ± 2.22	85.07 ± 3.16	**88.93±3.44**
User-3	320 (80)	88.03 ± 2.11	88.22 ± 2.56	**88.52±1.33**	88.19 ± 2.51
Avg		86.54	86.65	86.53	**88.26**

Table 3. Comparison on the MNIST datasets of (single-task) Adaboost, `MTL` and `MT-Adaboost`

Tasks	Train (Test)	Adaboost	`MTL` [15]	`MT-Adaboost`
1/-1	100 (10000)	91.770 ± 1.188	96.80 ± 1.91	96.802 ± 0.562
2/-2	100 (10000)	83.138 ± 2.347	69.95 ± 2.68	86.875 ± 0.676
3/-3	100 (10000)	82.959 ± 1.245	74.18 ± 5.54	87.679 ± 1.038
4/-4	100 (10000)	83.975 ± 1.408	71.76 ± 5.47	90.382 ± 0.713
5/-5	100 (10000)	78.423 ± 0.691	57.26 ± 2.72	84.253 ± 0.731
6/-6	100 (10000)	88.954 ± 1.601	80.54 ± 4.53	92.880 ± 0.896
7/-7	100 (10000)	87.105 ± 0.904	77.18 ± 9.43	92.811 ± 0.575
8/-8	100 (10000)	77.513 ± 1.905	65.85 ± 2.50	85.279 ± 1.727
9/-9	100 (10000)	81.842 ± 1.850	65.38 ± 6.09	86.904 ± 1.258
0/-0	300 (10000)	93.660 ± 1.287	97.81 ± 1.01	97.137 ± 0.418
Average	-	84.934	75.67	**90.100**

4.3 Results on Boosted Trees

In the previous section we experimentally validated the advantage of learning related tasks simultaneously, by using multi-task information gain criteria, in particular IG_M. In this section we compare boosted `MT-DT`'s to the boosted C4.5 trees. We use Adaboost.M1 [17] and MT-Adaboost (see algorithm 2) as boosters for C4.5 and for `MT-DT` respectively. Both algorithms have only one parameter, the number of boosting iterations which we set on a separated validation set.

Table 3 reports our comparison with the work in [15], we can see that our multi-task algorithm significantly outperform Adaboost with trees for single task learning and it proves a large margin of improvement over the method in [15], despite that we exactly follow their protocol.

Table 4. Average classification accuracy of boosted trees on Enron tasks

Tasks	Train (Test)	Adaboost C4.5	MT-Adaboost IG_J	MT-Adaboost IG_U	MT-Adaboost IG_M
Responsive Vs. NonResponsive	299 (74)	85.10 ± 1.21	84.66 ± 2.15	84.52 ± 1.2	**86.01±1.53**
5 Topics	265 (66)	51.34 ± 0.43	52.89 ± 0.87	52.17 ± 0.74	**57.11±0.02**
Avg		68.22	68.78	68.35	**71.65**

Table 4 reports the average values of classification accuracy over three random runs for Enron dataset. With boosted trees we observe an accuracy improvement similar to simple trees. Namely, MT-Adaboost+MT-DT is significantly better than Adaboost+C4.5; also the most difficult tasks enjoy a larger margin of improvement.

5 Conclusion

We proposed an adaptation of decision tree learning to the multi-task setting, with the following important contributions. First, we developed multi-task decision trees to deal with multi-class tasks with no label correspondence. The criterion to learn the decision rules makes use of the data from several tasks at each step of the decision tree learning, thus enabling to capture any degree of relatedness between the tasks. We then feature an important property of information gain rule when working with multiple tasks. This enabled us derive the new information gain criterion for learning decision trees in the multi-task setting. We also modified MT-Adaboost to cope with multi-class problems. We finally validated the proposed methods by series of experiments on three cases of multi-task learning.

Acknowledgment. This work was partially supported by Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council, FEDER through the *CPER 2007-2013* and the LAMPADA project co-funded by the *French Association on Research - ANR.*

References

1. Archembeau, C., Guo, S., Zoeter, O.: Sparse Bayesian Multi-Task Learning. In: NIPS (2011)
2. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. In: Advances in Neural Information Processing Systems 19 (2007)
3. Caruana, R.: Multitask learning. Machine Learning 28, 41–75 (1997)
4. Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., Tseng, B.: Multi-task learning for boosting with application to web search ranking. In: Proc. 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1189–1198 (2010)
5. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: ICML 2007: Proceedings of the 24th International Conference on Machine Learning, pp. 193–200. ACM (2007)
6. Evgeniou, T., Pontil, M.: Regularized multi–task learning. In: KDD 2004: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 109–117. ACM (2004)
7. Faddoul, J.B., Chidlovskii, B., Torre, F., Gilleron, R.: Boosting multi-task weak learners with applications to textual and social data. In: Proceedings of the Ninth International Conference on Machine Learning and Applications (ICMLA), pp. 367–372 (2010)

8. Freno, A., Trentin, E., Gori, M.: Kernel-based hybrid random fields for nonparametric density estimation. In: ECAI, pp. 427–432 (2010)
9. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
10. Gray, R.M.: Entropy and Information Theory, 2nd edn. Springer (2011)
11. Hovelynck, M., Chidlovskii, B.: Multi-modality in one-class classification. In: Proceedings of the 19th International Conference on World Wide Web (WWW), pp. 441–450 (2010)
12. Liu, T.-Y., Yang, Y., Wan, H., Zeng, H.-J., Chen, Z., Ma, W.-Y.: Support vector machines classification with a very large-scale taxonomy. SIGKDD Explor. Newsl. 7(1), 36–43 (2005)
13. Mantrach, A., Renders, J.-M.: A Mailbox Search Engine Using Query Multi-modal Expansion and Community-Based Smoothing. In: Baeza-Yates, R., de Vries, A.P., Zaragoza, H., Cambazoglu, B.B., Murdock, V., Lempel, R., Silvestri, F. (eds.) ECIR 2012. LNCS, vol. 7224, pp. 576–577. Springer, Heidelberg (2012)
14. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22, 1345–1359 (2010)
15. Quadrianto, N., Smola, A., Caetano, T., Vishwanathan, S.V.N., Petterson, J.: Multitask learning without label correspondences. In: Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems (NIPS), pp. 1957–1965 (2010)
16. Ross Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
17. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37 (1999)
18. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with dirichlet process priors. J. Mach. Learn. Res. 8, 35–63 (2007)

Multi-Task Boosting by Exploiting Task Relationships

Yu Zhang and Dit-Yan Yeung

Hong Kong University of Science and Technology
{zhangyu,dyyeung}@cse.ust.hk

Abstract. Multi-task learning aims at improving the performance of one learning task with the help of other related tasks. It is particularly useful when each task has very limited labeled data. A central issue in multi-task learning is to learn and exploit the relationships between tasks. In this paper, we generalize boosting to the multi-task learning setting and propose a method called multi-task boosting (MTBoost). Different tasks in MTBoost share the same base learners but with different weights which are related to the estimated task relationships in each iteration. In MTBoost, unlike ordinary boosting methods, the base learners, weights and task covariances are learned together in an integrated fashion using an alternating optimization procedure. We conduct theoretical analysis on the convergence of MTBoost and also empirical analysis comparing it with several related methods.

1 Introduction

In many real-world applications, the amount of labeled data available in a single learning task is scarce but there exist multiple related tasks. Multi-task learning [1,2,3] exploits this scenario to improve the performance of one learning task with the help of other related tasks. This learning paradigm, which can date back to some research in psychology and cognitive science, is inspired by human learning ability in that people often apply the knowledge gained from previous learning tasks to help learn a new task. For example, if a person can play Chinese chess, then (s)he will learn to play chess more easily by transferring the knowledge gained from playing Chinese chess. Major advances have been made in multi-task learning over the past decade. Multi-layered feedforward neural network [1] is one of the earliest models for multi-task learning. The units of the hidden layer in a neural network represent the common features for data points from all tasks and each unit in the output layer usually corresponds to the output of one task. Besides multi-layered feedforward neural networks, multi-task feature learning [4,5] also learns common features for all tasks with the difference being that it is a regularized method. Different from these methods which learn common data representations, some methods aim to learn similar model parameters for different tasks, e.g., regularized multi-task support vector machine (SVM) [6] defines a new regularizer to enforce the SVM parameters for all tasks to be close to each other. Moreover, one widely used approach for multi-task learning is the task clustering approach [7,8,9] in which the main idea is to group the tasks into several clusters and then learn identical or similar data features or model parameters for the tasks within each cluster. An advantage of this approach over the above mentioned methods is its robustness against outlier

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 697–710, 2012.

tasks because they reside in separate clusters that do not affect other tasks. Among many existing methods proposed for multi-task learning, a central issue in multi-task learning is to learn and exploit the pairwise relationships between tasks. There are three types of pairwise task relationships, namely, positive task correlation, negative task correlation, and task unrelatedness. However, most existing multi-task learning methods cannot make full use of all three types of task relationships. One way to incorporate the task relationships into a learning model is by adopting some model assumption about task relatedness. Unfortunately, the model assumption adopted may be incorrect. Worse still, it is not easy to verify the correctness of the model assumption. As such, it is more desirable to take an alternative approach by learning the task relationships from data automatically. The multi-task Gaussian process (GP) model [10] and its extension [11] are recently proposed methods that adopt this approach under the Bayesian framework. Moreover, Zhang and Yeung proposed a method in [12] to learn task relationships under the regularization framework for classification and regression problems, and then extended it for feature selection problems in [13].

Boosting [14], which seeks to combine multiple (weak) base learners to form a learner with significantly better performance, has been widely used in many areas, such as machine learning and data mining. There exist some explanations, e.g., margin theory [15,16], for the success of boosting. Moreover, some studies show that boosting is related to the additive model [17,18] in statistics. Even though boosting methods have shown good performance in many applications, their performance is often unsatisfactory when the labeled data is scarce. This calls for combining boosting and multi-task learning [19,20].

In this paper, we generalize boosting to the multi-task learning setting via learning and exploiting the pairwise task relationships. Our point of departure is a regularized method in [12] which learns the task relationships in the form of a task covariance matrix under a regularization framework and is related to maximum a posteriori (MAP) estimation of the weight-space interpretation of the multi-task GP model [10] presented in [11]. We then extend the formulation for boosting to give a method called multi-task boosting (MTBoost). By viewing boosting as a feature generating process, different tasks in MTBoost share the same base learners but with different weights which are related to the estimated task relationships in each iteration. Unlike single-task boosting methods which learn the base learners and weights separately, the base learners, weights and task covariances in MTBoost are learned together in an integrated fashion using an alternating optimization procedure. We conduct theoretical analysis on the convergence of the MTBoost learning algorithm.

The remainder of this paper is organized as follows. We present our MTBoost model and its learning algorithm in Section 2. Section 3 reviews some related work and Section 4 reports experimental results on some multi-task learning applications. Concluding remarks are given in the final section.

2 Multi-Task Boosting by Exploiting Task Relationships

Let there be m learning tasks $\{T_i\}_{i=1}^m$. For the ith task T_i, the training set consists of n_i labeled data points $(\mathbf{x}_j^i, y_j^i)$, $j = 1, \ldots, n_i$, with $\mathbf{x}_j^i \in \mathbb{R}^d$ and its corresponding output $y_j^i \in \{-1, 1\}$ for a binary classification problem.

2.1 Task Covariance Matrix

In [10], Bonilla et al. proposed a multi-task GP model which models the pairwise task relationships using a task covariance matrix under the Bayesian framework. However, it is not clear how to learn the task covariance matrix for other models such as those formulated under the regularization framework. In [12], the authors presented a regularized method, shedding light on how the task covariance matrix affects the learning of multiple tasks. More specifically, the task covariance matrix $\boldsymbol{\Omega}$ is used as a parameter matrix for the matrix-variate normal distribution [21] over the model parameters in least squares regression or support vector machine (SVM):

$$\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_m) \sim \mathcal{MN}_{d\times m}(\mathbf{A} \,|\, \mathbf{0}_{d\times m}, \mathbf{I}_d \otimes \boldsymbol{\Omega}), \tag{1}$$

where $\mathbf{a}_i \in \mathbb{R}^d$ is the model parameter vector for the ith task, $\mathcal{MN}_{d\times m}(\mathbf{M}, \boldsymbol{\Sigma} \otimes \boldsymbol{\Omega})$ denotes a matrix-variate normal distribution with mean $\mathbf{M} \in \mathbb{R}^{d\times m}$, row covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ and column covariance matrix $\boldsymbol{\Omega} \in \mathbb{R}^{m\times m}$. The probability density function of the matrix-variate normal distribution is

$$p(\mathbf{X} \,|\, \mathbf{M}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) = \frac{\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}(\mathbf{X}-\mathbf{M})\boldsymbol{\Omega}^{-1}(\mathbf{X}-\mathbf{M})^T\right)\right)}{(2\pi)^{md/2}|\boldsymbol{\Sigma}|^{m/2}|\boldsymbol{\Omega}|^{d/2}},$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a square matrix, $|\cdot|$ denotes the determinant of a square matrix, and $\mathbf{B}^{-1}$ denotes the inverse of a non-singular matrix $\mathbf{B}$ or the pseudo-inverse when it is singular. From this view, we can see that $\boldsymbol{\Omega}$ is used to model the covariance between the columns in the model parameter matrix $\mathbf{A}$ and hence to model the task relationships since each column in $\mathbf{A}$ represents the model parameters of the corresponding task.

Given the likelihood (i.e., logistic model for classification problem or Gaussian noise model for regression problem) and the prior defined in Eq. (1), the MAP solution is obtained by solving the following problem:

$$\begin{aligned} \min_{\mathbf{A},\boldsymbol{\Omega}} \; & \sum_{i=1}^{m}\sum_{j=1}^{n_i} l(\mathbf{a}_i^T\mathbf{x}_j^i, y_j^i) + \frac{\lambda}{2}\mathrm{tr}(\mathbf{A}\boldsymbol{\Omega}^{-1}\mathbf{A}^T) \\ \text{s.t.} \; & \boldsymbol{\Omega} \succeq 0, \; \mathrm{tr}(\boldsymbol{\Omega}) = 1, \end{aligned} \tag{2}$$

where $l(\cdot,\cdot)$ defines the empirical loss corresponding to the likelihood, λ is a regularization parameter which balances the tradeoff between the empirical loss and the regularization term, and $\boldsymbol{\Omega} \succeq 0$ means that the matrix $\boldsymbol{\Omega}$ is positive semidefinite. The first constraint in problem (2) is needed because $\boldsymbol{\Omega}$ is defined as a task covariance matrix and the second constraint serves to restrict the complexity of $\boldsymbol{\Omega}$. The second term in the objective function of problem (2) is derived from the matrix-variate normal prior in Eq. (1) and is used to regularize the task relationships.

It is easy to show that problem (2) is a convex problem as long as the loss function $l(\cdot,\cdot)$ is convex, as proved in [12]. We will show how to design a boosting algorithm according to problem (2).

2.2 Multi-Task Boosting

In a boosting algorithm, we are given a fixed class of functions (or called base hypotheses) denoted by $\mathcal{F}$ and are required to find a linear combination of functions in $\mathcal{F}$, denoted by $\text{lin}(\mathcal{F})$, that minimizes a cost functional $C(\cdot)$ on $\text{lin}(\mathcal{F})$. For example, in our experiments, due to the high-dimensional data involved, a linear least-squares SVM is utilized as the base learner. The final hypothesis can be written as $F(\mathbf{x}) = \sum_{t=1}^{Q} w_t f_t(\mathbf{x})$ where $f_t \in \mathcal{F}$ and $w_i \in \mathbb{R}$. So, boosting may be viewed as finding for each data point $\mathbf{x}$ a new feature representation $\mathbf{z} = (f_1(\mathbf{x}), \ldots, f_Q(\mathbf{x}))^T$ and also a coefficient vector. From this view, we can extend problem (2) for multi-task boosting as

$$\begin{aligned} \min_{\mathbf{W}, \boldsymbol{\Omega}, \{f_t\}} \quad & \sum_{i=1}^{m} \sum_{j=1}^{n_i} l(\mathbf{w}_i^T \mathbf{z}_j^i, y_j^i) + \frac{\lambda}{2} \text{tr}(\mathbf{W} \boldsymbol{\Omega}^{-1} \mathbf{W}^T) \\ \text{s.t.} \quad & \mathbf{z}_j^i = (f_1(\mathbf{x}_j^i), \ldots, f_Q(\mathbf{x}_j^i))^T \ \forall i, j \\ & \boldsymbol{\Omega} \succeq 0, \ \text{tr}(\boldsymbol{\Omega}) = 1, \end{aligned} \tag{3}$$

where $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$. In this formulation, we can see that the final hypothesis F_i for the ith task can be expressed as $F_i = \sum_{j=1}^{Q} w_{ij} f_j$, where w_{ij} is the jth element of $\mathbf{w}_i$. So different tasks share the same base hypotheses but with different weights.

However, here we cannot know the base hypotheses in advance. So we view this model as an additive model [17] and use the gradient boosting technique [18,22] to learn the base hypotheses and their weights. More specifically, in the t-th iteration, the existing combined hypothesis for the ith task is denoted by $F_i^{(t-1)}$ and the optimization problem is

$$\begin{aligned} \min_{\mathbf{w}^t, \boldsymbol{\Omega}, f_t} \quad & \sum_{i=1}^{m} C(F_i^{(t-1)} + w_{it} f_t) + \frac{\lambda}{2} \text{tr}(\mathbf{W}_t \boldsymbol{\Omega}^{-1} \mathbf{W}_t^T) \\ \text{s.t.} \quad & \boldsymbol{\Omega} \succeq 0, \ \text{tr}(\boldsymbol{\Omega}) = 1, \end{aligned} \tag{4}$$

where $\mathbf{W}_t$ is the weight matrix until the tth iteration with (i, j)th element as w_{ji} and $\mathbf{w}^t = (w_{1t}, \ldots, w_{mt})$ denotes the new weight vector obtained in the tth iteration. Here $C(F_i) = \sum_{j=1}^{n_i} l(F_i(\mathbf{x}_j^i), y_j^i)$. Since we mainly consider the classification problem in this section, we take the margin cost functional to be the loss function, i.e., $l(F(\mathbf{x}_j^i), y_j^i) = c(y_j^i F(\mathbf{x}_j^i))$ for some monotonically decreasing function $c(\cdot)$. In this paper, $c(\cdot)$ takes the form of $c(x) = \ln(1 + \exp(-x))$ which is widely used in boosting algorithms such as LogitBoost [17].

Since problem (4) is still not easy to solve, we use the majorization-minimization (MM) algorithm [23] to solve it. The MM algorithm is an iterative algorithm which seeks an upper bound of the objective function based on the solution of the previous iteration as a surrogate function for a minimization problem and minimizes the surrogate function instead of the original objective function. It has been proved that the MM algorithm is guaranteed to find a local optimum. Here for simplicity, we just run one iteration of the MM algorithm with the initial solution of f_t as a zero function. Since $c(x) = \ln(1+\exp(-x))$

is a concave function due to the fact that $\frac{\partial^2 c(x)}{\partial^2 x} = -\frac{\exp(-x)}{(1+\exp(-x))^2} < 0$, $C(\cdot)$ is also a concave functional and hence we have

$$C(F_i^{(t-1)} + w_{it} f_t) \leq C(F_i^{(t-1)}) + w_{it} \langle \nabla C(F_i^{(t-1)}), f_t \rangle,$$

due to the first-order property of a concave function. Here $\nabla C(F_i)$ denotes the functional derivative of C at F_i and $\langle F_i, G_i \rangle$ is the inner product which is defined as $\langle F_i, G_i \rangle = \sum_{j=1}^{n_i} F_i(\mathbf{x}_j^i) G_i(\mathbf{x}_j^i)$. So in each iteration of the MM algorithm, the optimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{w}^t, \boldsymbol{\Omega}, f_t} \ & \sum_{i=1}^{m} w_{it} \langle \nabla C(F_i^{(t-1)}), f_t \rangle + \frac{\lambda}{2} \mathrm{tr}(\mathbf{W}_t \boldsymbol{\Omega}^{-1} \mathbf{W}_t^T) \\ \text{s.t.} \ & \boldsymbol{\Omega} \succeq 0, \ \mathrm{tr}(\boldsymbol{\Omega}) = 1. \end{aligned} \tag{5}$$

Unlike conventional boosting algorithms which can optimize w_{it} and f_t separately as in [18, 22], here in problem (5) w_{it} and f_t are coupled together. We use an alternating method to solve the problem.

When $\mathbf{w}^t$ and $\boldsymbol{\Omega}$ are given, we can get

$$\langle \nabla C(F_i^{(t-1)}), f_t \rangle = \sum_{j=1}^{n_i} y_j^i f_t(\mathbf{x}_j^i) c'(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i)),$$

where $c'(\cdot)$ is the derivative of $c(\cdot)$, since $C(F_i^{(t-1)}) = \sum_{j=1}^{n_i} c(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i))$. Then we need to solve the following minimization problem to find f_t:

$$\min_{f_t} \sum_{i=1}^{m} w_{it} \sum_{j=1}^{n_i} y_j^i f_t(\mathbf{x}_j^i) c'(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i)). \tag{6}$$

Since $c(\cdot)$ is monotonically decreasing, the derivative $c'(\cdot)$ is negative. Problem (6) can be reformulated as

$$\max_{f_t} \sum_{i=1}^{m} \sum_{j=1}^{n_i} \tilde{y}_j^i f_t(\mathbf{x}_j^i) d_j^i, \tag{7}$$

where $d_j^i = \frac{c'(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i))|w_{it}|}{\sum_{i=1}^{m} |w_{it}| \sum_{j=1}^{n_i} c'(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i))}$ defines the instance weight for $\mathbf{x}_j^i$, $\tilde{y}_j^i = \mathrm{sign}(w_{it}) y_j^i$, and $\mathrm{sign}(\cdot)$ is the sign function. Since $w_{it} \in \mathbb{R}$, here we take the absolute value of w_{it} to keep the instance weights $\{d_j^i\}$ non-negative and the equivalence between problem (6) and (7) is due to the fact that $w_{it} = \mathrm{sign}(w_{it})|w_{it}|$. We assume $f_t(\cdot) \in \{-1, 1\}$. Since $\tilde{y}_j^i \in \{-1, 1\}$,[1] the objective function in problem (7) can be rewritten as

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} \tilde{y}_j^i f_t(\mathbf{x}_j^i) d_j^i = \sum_{\tilde{y}_j^i = f_t(\mathbf{x}_j^i)} d_j^i - \sum_{\tilde{y}_j^i \neq f_t(\mathbf{x}_j^i)} d_j^i = 1 - 2 \sum_{\tilde{y}_j^i \neq f_t(\mathbf{x}_j^i)} d_j^i.$$

[1] When $w_{it} = 0$, the ith task has no effect on problem (7) and hence can be ignored.

So problem (7) is equivalent to minimizing the weighted classification error

$$\min_{f_t} \sum_{\tilde{y}_j^i \neq f_t(\mathbf{x}_j^i)} d_j^i. \tag{8}$$

From problem (8), we may look at it as a weighted combination of multiple tasks to give a single "supertask" with possible label flipping from y_j^i to $\tilde{y}_j^i$ depending on the relationships between tasks. For a base learner such as least-squares SVM, we need to solve a weighted least-squares SVM where the instance weights are defined by $\{d_j^i\}$.

When f_t and $\mathbf{w}^t$ are given, we need to solve the following problem

$$\begin{aligned} \min_{\boldsymbol{\Omega}} \ & \mathrm{tr}(\mathbf{W}_t \boldsymbol{\Omega}^{-1} \mathbf{W}_t^T) \\ \text{s.t.} \ & \boldsymbol{\Omega} \succeq 0, \ \mathrm{tr}(\boldsymbol{\Omega}) = 1. \end{aligned} \tag{9}$$

Then we have

$$\begin{aligned} \mathrm{tr}(\boldsymbol{\Omega}^{-1}\mathbf{B}) &= \mathrm{tr}(\boldsymbol{\Omega}^{-1}\mathbf{B})\mathrm{tr}(\boldsymbol{\Omega}) \\ &= \mathrm{tr}((\boldsymbol{\Omega}^{-\frac{1}{2}}\mathbf{B}^{\frac{1}{2}})(\mathbf{B}^{\frac{1}{2}}\boldsymbol{\Omega}^{-\frac{1}{2}}))\mathrm{tr}(\boldsymbol{\Omega}^{\frac{1}{2}}\boldsymbol{\Omega}^{\frac{1}{2}}) \\ &\geq (\mathrm{tr}(\boldsymbol{\Omega}^{-\frac{1}{2}}\mathbf{B}^{\frac{1}{2}}\boldsymbol{\Omega}^{\frac{1}{2}}))^2 = (\mathrm{tr}(\mathbf{B}^{\frac{1}{2}}))^2, \end{aligned}$$

where $\mathbf{B} = \mathbf{W}_t^T\mathbf{W}_t$. The first equality holds because of the last constraint in problem (9), and the last inequality holds because of the Cauchy-Schwarz inequality for the Frobenius norm. Moreover, $\mathrm{tr}(\boldsymbol{\Omega}^{-1}\mathbf{B})$ attains its minimum value $(\mathrm{tr}(\mathbf{B}^{\frac{1}{2}}))^2$ if and only if $\boldsymbol{\Omega}^{-\frac{1}{2}}\mathbf{B}^{\frac{1}{2}} = a\boldsymbol{\Omega}^{\frac{1}{2}}$ for some constant a and $\mathrm{tr}(\boldsymbol{\Omega}) = 1$. So we can get the analytical solution

$$\boldsymbol{\Omega} = \frac{(\mathbf{W}_t^T\mathbf{W}_t)^{\frac{1}{2}}}{\mathrm{tr}\left((\mathbf{W}_t^T\mathbf{W}_t)^{\frac{1}{2}}\right)}. \tag{10}$$

When f_t and $\boldsymbol{\Omega}$ are given, the optimization problem for $\mathbf{w}^t$ is formulated as

$$\min_{\mathbf{w}^t} J(\mathbf{w}^t) = \mathbf{w}^t\boldsymbol{\beta}_t + \frac{\lambda}{2}\mathbf{w}^t\boldsymbol{\Omega}^{-1}(\mathbf{w}^t)^T, \tag{11}$$

where $\boldsymbol{\beta}_t = (\langle \nabla C(F_1^{(t-1)}), f_t\rangle, \ldots, \langle \nabla C(F_m^{(t-1)}), f_t\rangle)^T$. We set the derivative of problem (11) with respect to $\mathbf{w}^t$ to zero to get the solution of $\mathbf{w}^t$ as

$$\mathbf{w}^t = -\frac{1}{\lambda}(\boldsymbol{\beta}_t)^T\boldsymbol{\Omega}. \tag{12}$$

We summarize the MTBoost algorithm in Table 1.

For the initialization of $\mathbf{w}^t$, we randomly generate it from a normal distribution with zero mean and $\boldsymbol{\Omega}$ as the covariance matrix due to the matrix-variate normal prior on $\mathbf{W}_t$ in Eq. (1).

The whole procedure of our MTBoost algorithm includes solving problem (8) and updating $\boldsymbol{\Omega}$ and $\mathbf{w}^t$ according to Eqs (10) and (12). The main computational cost lies in solving problem (8) whose complexity equals the computational cost of training a base learner (i.e., SVM or least-squares SVM) on the training data of all tasks.

Table 1. Algorithm for Multi-Task Boosting (MTBoost)

Input: $\{(\mathbf{x}_j^i, y_j^i)\}_{j=1}^{n_i}$ $(i = 1, \ldots, m)$, λ, maximum numbers of iterations Q and Q_1
Let $F_i^{(0)}(x) := 0$ for $i = 1, \ldots, m$ and $\boldsymbol{\Omega} := \mathbf{I}_m$;
for $t := 1$ to Q do
Initialize $\mathbf{w}^t$;
for $t_1 := 1$ to Q_1
Update f_t by solving problem (8);
Update $\boldsymbol{\Omega}$ via Eq. (10);
Update $\mathbf{w}^t$ via Eq. (12);
end for
Let $F_i^{(t)} := F_i^{(t-1)} + w_{it} f_t$ for $i = 1, \ldots, m$;
if $(\boldsymbol{\beta}_t)^T \boldsymbol{\Omega} \boldsymbol{\beta}_t \leq \varepsilon$
break;
end if
end for
Output: $F_i^{(Q)}$ for $i = 1, \ldots, m$

2.3 Theoretical Analysis

In this section, we prove the convergence of the MTBoost algorithm.

Theorem 1. *The solution minimizing problem (5) also minimizes problem (4).*

Proof Let $G(\mathbf{w}^t, \boldsymbol{\Omega}, f_t)$ denote the objective function of problem (4) and $H(\mathbf{w}^t, \boldsymbol{\Omega}, f_t)$ denote the objective function of problem (5). Due to the concavity of $C(\cdot)$, for any $\mathbf{w}^t$, $\boldsymbol{\Omega}$ and f_t, we have

$$G(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) \leq H(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) + \sum_{i=1}^{m} C(F_i^{(t-1)}).$$

Moreover, we have

$$G(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0) = H(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0) + \sum_{i=1}^{m} C(F_i^{(t-1)}),$$

where 0 denotes the zero function and $\mathbf{w}_0^t$ and $\boldsymbol{\Omega}_0$ are the initial values for the variables $\mathbf{w}^t$ and $\boldsymbol{\Omega}$. For the solution $(\mathbf{w}^t, \boldsymbol{\Omega}, f_t)$ minimizing problem (5), we have

$$H(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) \leq H(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0).$$

Then we can get

$$\begin{aligned} G(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) &\leq H(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) + \sum_{i=1}^{m} C(F_i^{(t-1)}) \\ &\leq H(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0) + \sum_{i=1}^{m} C(F_i^{(t-1)}) \\ &= G(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0), \end{aligned}$$

which means the value of the objective function of problem (4) at the solution of problem (5) is lower than that at the initial values. Hence we prove the result. □

Theorem 2.

$$\sum_{i=1}^{m} C(F_i^{(t)}) \leq \sum_{i=1}^{m} C(F_i^{(t-1)}) - \frac{1}{\lambda}\boldsymbol{\beta}_t \boldsymbol{\Omega} \boldsymbol{\beta}_t \leq \sum_{i=1}^{m} C(F_i^{(t-1)})$$

Proof Due to the concavity of $C(\cdot)$, we have

$$\begin{aligned} C(F_i^{(t)}) &= C(F_i^{(t-1)} + w_{it} f_t) \\ &\leq C(F_i^{(t-1)}) + w_{it} \langle \nabla C(F_i^{(t-1)}), f_t \rangle. \end{aligned}$$

Then we can get

$$\begin{aligned} \sum_{i=1}^{m} C(F_i^{(t)}) &\leq \sum_{i=1}^{m} C(F_i^{(t-1)}) + w_{it} \langle \nabla C(F_i^{(t-1)}), f_t \rangle \\ &= \sum_{i=1}^{m} C(F_i^{(t-1)}) + \mathbf{w}^t \boldsymbol{\beta}_t \\ &= \sum_{i=1}^{m} C(F_i^{(t-1)}) - \frac{1}{\lambda} (\boldsymbol{\beta}_t)^T \boldsymbol{\Omega} \boldsymbol{\beta}_t, \end{aligned}$$

where the last equality holds due to the relationship between $\mathbf{w}^t$ and $\boldsymbol{\beta}_t$ reflected in Eq. (12). Moreover, since $\boldsymbol{\Omega}$ is a positive semi-definite matrix which can be verified by the solution of $\boldsymbol{\Omega}$ in Eq. (10), we have

$$\frac{1}{\lambda} (\boldsymbol{\beta}_t)^T \boldsymbol{\Omega} \boldsymbol{\beta}_t \geq 0$$

and hence

$$\sum_{i=1}^{m} C(F_i^{(t-1)}) - \frac{1}{\lambda}\boldsymbol{\beta}_t \boldsymbol{\Omega} \boldsymbol{\beta}_t \leq \sum_{i=1}^{m} C(F_i^{(t-1)}).$$

Finally we reach the conclusion. □

From Theorem 2, we can see that when adding a new component classifier in MTBoost, the empirical loss of all tasks decreases. Since the empirical loss is non-negative, our method is guaranteed to converge. Moreover, Theorem 2 suggests a termination criterion for the MTBoost algorithm in Table 1: $(\boldsymbol{\beta}_t)^T \boldsymbol{\Omega} \boldsymbol{\beta}_t$ is small and below a threshold ε.

3 Related Work

Duchi and Singer [24] proposed a boosting method for multi-class classification problems by utilizing the structural sparsity of model parameters. They claimed that the method can be generalized for multi-task learning. An underlying assumption of their method is that all tasks are similar and they share a similar model or data representation.

However, in many applications, there exist tasks which exhibit negative task correlation or task unrelatedness and hence the assumption is violated, impairing the performance of the method.

Wang et al. [19] extended the idea of task clustering [8] to boosting by grouping the tasks into several clusters and learning similar data features or model parameters for the tasks within each cluster. This approach is robust against outlier tasks because outlier tasks reside in separate clusters that do not affect other tasks, but they are local methods in the sense that only similar tasks within the same task cluster can interact to help each other, thus ignoring negative task correlation which may exist between tasks residing in different clusters. Moreover, how to determine the number of clusters is a difficult model selection problem.

Chapelle et al. [20] proposed a multi-boost method for multi-task learning which assumes that the model parameters in different tasks are similar and utilizes the difference between different model parameters to define a regularization term for boosting. The relationship between the multi-boost method and single-task boosting is similar to that between regularized multi-task SVM [6] and single-task SVM. Moreover, similar to [24], the multi-boost method also uses l_1 regularization to enforce sparsity.

Dai et al. [25] proposed a boosting method for transfer learning. Transfer learning is related to multi-task learning but there exist some differences. The tasks in transfer learning can be divided into source and target tasks and transfer learning aims at improving the performance of the target task with the help of the source tasks while multi-task learning seeks to improve the performance of all tasks simultaneously. Pardoe and Stone [26] extended boosting to regression problems in the transfer setting.

4 Experiment

In this section, we study MTBoost empirically on some applications and compare it with a single-task boosting method called AnyBoost [22], a multi-task boosting method called Multi-boost [20], a multi-task learning method called multi-task GP (MTGP) [10] and MTRL [12] which can also learn the task relationships under the GP and regularization framework respectively.[2]

4.1 Multi-domain Sentiment Classification

In this subsection, we study a multi-domain sentiment classification application[3] which is naturally a multi-task classification problem. Its goal is to classify the reviews of some products into two classes: positive and negative reviews. In this application, there are four different products (tasks) from Amazon.com: books, DVDs, electronics, and kitchen appliances. For each task, there are 1,000 positive and 1,000 negative data points corresponding to positive and negative reviews, respectively. Each data point has 473,856 feature dimensions.

[2] The implementation of our method can be downloaded from `http://www.cse.ust.hk/~dyyeung/code/MTBoost.zip`

[3] `http://www.cs.jhu.edu/~mdredze/datasets/sentiment/`

Table 2. Comparison of different methods on multi-domain sentiment classification. Each column in the table represents one task. For each method, the first row records the mean classification error over 10 trials and the second row records the standard deviation. 1st task: books; 2nd task: DVDs; 3rd task: electronics; 4th task: kitchen appliances.

Method	1st Task	2nd Task	3rd Task	4th Task
AnyBoost	0.2595	0.2500	0.1999	0.1789
	0.0086	0.0085	0.0096	0.0054
MTGP	0.2594	0.2510	0.2493	0.2407
	0.0097	0.0089	0.0076	0.0085
Multi-boost	0.2918	0.3041	0.3116	0.3165
	0.0138	0.0122	0.0204	0.0175
MTRL	0.2474	**0.2233**	0.1925	0.1719
	0.0116	0.0115	0.0135	0.0098
MTBoost	**0.2385**	**0.2236**	**0.1666**	**0.1491**
	0.0105	0.0087	0.0054	0.0114

Since the feature dimensionality is very high making tree classifiers such as C4.5 and decision stump difficult to use, we choose linear least-squares SVM as the base learner in AnyBoost, Multi-boost and our MTBoost method. To simulate real applications in which the labeled data is scarce, we choose only 20% of the data in each task to form the training set and the rest to form the test set. We perform 10 random splits of the data and report the mean and standard deviation over the 10 trials. The number of rounds in AnyBoost, Multi-Boost and MTBoost is set to 100 and the number of the inner iterations of our MTBoost (i.e., Q_1) is set to 10. The optimal λ is determined by 5-fold cross validation where the candidate set is $\{0.01, 0.1, 1, 10, 100\}$. The results are summarized in Table 2 and the best result after pairwise t-test is shown in bold. From the table, we can see that MTBoost outperforms AnyBoost, Multi-boost, MTGP and MTRL on almost every task. Moreover, we notice that the performance of Multi-boost is just comparable or even worse than that of AnyBoost. One possible reason is that not all the tasks are very similar to each other as can be revealed by the mean task correlation matrix shown in Table 3. In other words, the assumption underlying Multi-boost is not satisfied well in this data set.

The mean task correlation matrix over 10 trials is shown in Table 3. We can see that the first task 'books' is more correlated with the second task 'DVDs' than with the other tasks; the third and fourth tasks achieve the highest correlation among all pairs of tasks. The finding from Table 3 about the relationships between tasks matches our intuition, with the following possible interpretation: 'books' and 'DVDs' are mainly for entertainment; and almost all the elements in 'kitchen appliances' belong to 'electronics'.

4.2 Handwritten Letter Classification

The handwritten letter classification applicaton[4] consists of seven tasks each of which is a binary classification problem. The corresponding letter pairs for the seven tasks are:

[4] http://multitask.cs.berkeley.edu/

Table 3. Mean task correlation matrix over 10 trials for multi-domain sentiment data. 1st task: books; 2nd task: DVDs; 3rd task: electronics; 4th task: kitchen appliances.

	1st	2nd	3rd	4th
1st	1.0000	0.6977	0.6253	0.6357
2nd	0.6977	1.0000	0.6306	0.6186
3rd	0.6253	0.6306	1.0000	0.7994
4th	0.6357	0.6186	0.7994	1.0000

Table 4. Comparison of different methods on handwritten letter classification. Each column in the table represents one task. For each method, the first row records the mean classification error over 10 trials and the second row records the standard deviation.

Method	1st Task	2nd Task	3rd Task	4th Task	5th Task	6th Task	7th Task
AnyBoost	0.1330	0.3026	0.1271	0.0970	0.0895	0.1997	0.0689
	0.0231	0.0135	0.0311	0.0068	0.0137	0.0178	0.0062
MTGP	0.1316	0.2844	0.1146	0.0903	0.1349	0.2177	0.0852
	0.0135	0.0070	0.0153	0.0075	0.0256	0.0388	0.0456
Multi-boost	0.2064	0.3425	0.2144	0.1295	0.1301	0.2111	0.0989
	0.0548	0.1238	0.1001	0.0675	0.0175	0.0327	0.0754
MTRL	0.1184	0.2790	0.1058	0.0824	0.0880	0.2180	**0.0561**
	0.0048	0.0128	0.0090	0.0060	0.0057	0.0131	0.0073
MTBoost	**0.1136**	**0.2642**	**0.1001**	**0.0611**	**0.0830**	**0.1924**	0.0623
	0.0161	0.0108	0.0127	0.0066	0.0105	0.0233	0.0078

Table 5. Comparison of different methods on USPS digit classification. Each column in the table represents one task. For each method, the first row records the mean classification error over 10 trials and the second row records the standard deviation.

Method	1st Task	2nd Task	3rd Task	4th Task	5th Task	6th Task	7th Task	8th Task	9th Task
AnyBoost	0.0040	0.0120	0.0437	0.0166	0.0292	0.0490	0.0078	0.0234	0.0232
	0.0003	0.0017	0.0034	0.0028	0.0076	0.0080	0.0027	0.0090	0.0066
MTGP	0.0009	0.0050	0.0374	0.0139	**0.0252**	0.0484	0.0069	0.0232	0.0230
	0.0005	0.0028	0.0035	0.0025	0.0073	0.0065	0.0034	0.0085	0.0087
Multi-boost	0.0010	0.0107	0.0421	0.0153	0.0276	0.0481	0.0065	0.0249	0.0259
	0.0363	0.0082	0.0088	0.0037	0.0021	0.0085	0.0041	0.0078	0.0078
MTRL	0.0008	0.0048	0.0353	0.0120	0.0267	**0.0340**	**0.0029**	0.0225	0.0223
	0.0007	0.0011	0.0035	0.0044	0.0014	0.0040	0.0019	0.0026	0.0046
MTBoost	**0.0003**	**0.0040**	**0.0324**	**0.0105**	**0.0252**	**0.0348**	0.0052	**0.0206**	**0.0213**
	0.0005	0.0034	0.0056	0.0042	0.0094	0.0075	0.0017	0.0060	0.0095

c/e, g/y, m/n, a/g, a/o, f/t and h/n. Each data point has 128 features corresponding to the pixel values of the handwritten letter images. For each task, there are about 1,000 positive and 1,000 negative data points. The experimental settings are the same as those for the multi-domain sentiment application above.

The mean classification errors and the standard deviations of different methods over 10 trials are summarized in Table 4. The results show that MTBoost outperforms AnyBoost on every task, showing the effectiveness of sharing between multiple learning tasks. MTGP, another method which can learn the task covariance matrix from data, performs better than AnyBoost on some tasks but worse on other tasks. One possible reason is that MTGP usually uses low-rank approximation for the task covariance matrix to reduce the computational cost. This may affect the expressive power of the model and impair its performance. Moreover, MTBoost outperforms Multi-boost, MTGP and MTRL.

4.3 USPS Digit Classification

The USPS digit data set[5] contains 7,291 examples each of which is described by 255 features. There are nine classification tasks, each corresponding to the classification of two digits. The experimental settings are similar to those in the above subsections. The mean classification errors and the standard deviations of different methods over 10 trials are summarized in Table 5. Again, we find that MTBoost outperforms AnyBoost, Multi-boost and MTGP on almost every task. Moreover, MTBoost performs better than MTRL on most tasks and comparable with MTRL on the other tasks.

5 Conclusion

In this paper, we have proposed a multi-task boosting method based on learning and exploiting the pairwise relationships between tasks. The alternating optimization procedure in MTBoost has been shown to converge with theoretical guarantee. In the current setting, each task is a binary classification problem. One future direction is to extend this work to a general setting where each task can be either a binary or a multi-class classification problem. Besides, we mainly consider classification problems in this paper. In our future work, we will also investigate the extension of our method to regression problems. One possibility is to make use of the loss function defined in [27] for such problems.

Acknowledgment. This research has been supported by General Research Fund 621310 from the Research Grants Council of Hong Kong.

References

1. Caruana, R.: Multitask learning. Machine Learning 28(1), 41–75 (1997)
2. Baxter, J.: A Bayesian/information theoretic model of learning to learn via multiple task sampling. Machine Learning 28(1), 7–39 (1997)

[5] http://multitask.cs.berkeley.edu/

3. Thrun, S.: Is learning the n-th thing any easier than learning the first? In: Touretzky, D.S., Mozer, M., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems 8, Denver, CO, pp. 640–646 (1996)
4. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) Advances in Neural Information Processing Systems 20, Vancouver, British Columbia, Canada, pp. 41–48 (2007)
5. Argyriou, A., Evgeniou, T., Pontil, M.: Convex multi-task feature learning. Machine Learning 73(3), 243–272 (2008)
6. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, pp. 109–117 (2004)
7. Thrun, S., O'Sullivan, J.: Discovering structure in multiple learning tasks: The TC algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, Bari, Italy, pp. 489–497 (1996)
8. Bakker, B., Heskes, T.: Task clustering and gating for bayesian multitask learning. Journal of Machine Learning Research 4, 83–99 (2003)
9. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with Dirichlet process priors. Journal of Machine Learning Research 8, 35–63 (2007)
10. Bonilla, E., Chai, K.M.A., Williams, C.: Multi-task Gaussian process prediction. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems 20, Vancouver, British Columbia, Canada, pp. 153–160 (2008)
11. Zhang, Y., Yeung, D.Y.: Multi-task learning using generalized t process. In: Proceedings of the 13rd International Conference on Artificial Intelligence and Statistics, Chia Laguna Resort, Sardinia, Italy, pp. 964–971 (2010)
12. Zhang, Y., Yeung, D.Y.: A convex formulation for learning task relationships in multi-task learning. In: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, Catalina Island, California, pp. 733–742 (2010)
13. Zhang, Y., Yeung, D.Y., Xu, Q.: Probabilistic multi-task feature selection. In: Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Advances in Neural Information Processing Systems 23, Vancouver, British Columbia, Canada, pp. 2559–2567 (2010)
14. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the 13th International Conference on Machine Learning, Bari, Italy, pp. 148–156 (1996)
15. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics 26(5), 1651–1686 (1998)
16. Reyzin, L., Schapire, R.E.: How boosting the margin can also boost classifier complexity. In: Proceedings of the Twenty-Third International Conference on Machine Learning, Pittsburgh, Pennsylvania, USA, pp. 753–760 (2006)
17. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. The Annals of Statistics 28(2), 337–407 (2000)
18. Friedman, J.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics 29(5), 1189–1232 (2001)
19. Wang, X., Zhang, C., Zhang, Z.: Boosted multi-task learning for face verification with applications to web image and video search. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Miami, Florida, USA, pp. 142–149 (2009)

20. Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., Tseng, B.: Multi-task learning for boosting with application to web search ranking. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, pp. 1189–1198 (2010)
21. Gupta, A.K., Nagar, D.K.: Matrix Variate Distributions. Chapman & Hall (2000)
22. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent. In: Solla, S.A., Leen, T.K., Müller, K.R. (eds.) Advances in Neural Information Processing Systems 12, Denver, Colorado, USA, pp. 512–518 (1999)
23. Lange, K., Hunter, D.R., Yang, I.: Optimization transfer using surrogate objective functions. Journal of Computational and Graphical Statistics 9(1), 1–59 (2000)
24. Duchi, J., Singer, Y.: Boosting with structural sparsity. In: Proceedings of the 26th International Conference on Machine Learning, Montreal, Quebec, Canada, pp. 297–304 (2009)
25. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, USA, pp. 193–200 (2007)
26. Pardoe, D., Stone, P.: Boosting for regression transfer. In: Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, pp. 863–870 (2010)
27. Zemel, R.S., Pitassi, T.: A gradient-based boosting algorithm for regression problems. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems 13, Denver, CO, USA, pp. 696–702 (2000)

Sparse Gaussian Processes for Multi-task Learning

Yuyang Wang and Roni Khardon

Tufts University, Medford, MA USA
{ywang02,roni}@cs.tufts.edu

Abstract. Multi-task learning models using Gaussian processes (GP) have been recently developed and successfully applied in various applications. The main difficulty with this approach is the computational cost of inference using the union of examples from all tasks. The paper investigates this problem for the grouped mixed-effect GP model where each individual response is given by a fixed-effect, taken from one of a set of unknown groups, plus a random individual effect function that captures variations among individuals. Such models have been widely used in previous work but no sparse solutions have been developed. The paper presents the first sparse solution for such problems, showing how the sparse approximation can be obtained by maximizing a variational lower bound on the marginal likelihood, generalizing ideas from single-task Gaussian processes to handle the mixed-effect model as well as grouping. Experiments using artificial and real data validate the approach showing that it can recover the performance of inference with the full sample, that it outperforms baseline methods, and that it outperforms state of the art sparse solutions for other multi-task GP formulations.

1 Introduction

In multi-task learning one learns multiple related tasks simultaneously, with the intention of getting improved predictive performance for all tasks by taking advantage of the common aspects of the tasks. In this paper we explore Bayesian models especially using Gaussian Processes (GP) where sharing the prior and its parameters among the tasks can be seen to implement multi-task learning [3, 4, 18, 7]. Our focus is on the functional grouped mixed-effect model [5, 17] where each task is modeled as a sum of a group-specific fixed-effect (or mean effect, group effect) shared by all the tasks in the group and a random effect that can be interpreted as representing task specific deviations. In particular, all effects are realizations of zero-mean Gaussian processes. Thus, in this model, tasks share structure through hyper-parameters of the prior and through the group-specific fixed-effect portion. This model and its single center counterpart [5, 7] (the classical mixed-effect GP model) have shown success in a wide range of applications, including geophysics [5], medicine [7] and astrophysics [17]. One of the main difficulties with this model, however, is the computational cost, because while the number of samples per task N_j is small, the total sample size

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 711–727, 2012.

$\sum_j N_j$ can be large, and the typical cubic complexity of GP inference can be prohibitively large [18]. Some improvement can be obtained when all the tasks share the same sampling points, or when different tasks share many of the input points [6, 7]. However, if the number of distinct sampling points is large the complexity remains high. For example, this is the case in [17] where sample points are clipped to a fine grid to avoid the high cardinality of the example set.

The same problem, handling large samples, has been extensively studied in single task formalizations of GP, where several approaches for so-called *sparse solutions* have been developed [11–13, 15]. These methods approximate the GP with $m \ll N$ support variables (also called inducing variables or pseudo inputs) $\mathcal{X}_m$ and their corresponding function values $\mathbf{f}_m$ and perform inference using this set. In the multi-task GP literature, sparse solutions have been proposed in [4] and [1] for a multi-task GP formulation that is different from the one considered in this paper. A more detailed discussion is given in Section 5.

In this paper, we develop a sparse solution for multi-task learning with GP in the context of the functional grouped mixed-effect model. Specifically, we extend the approach of [15] and develop a variational approximation that allows us to efficiently learn the shared hyper-parameters and choose the support variables. In addition, we show how the variational approximation can be used to perform prediction efficiently once learning has been performed. Our approach is particularly useful *when individual tasks have a small number of samples, different tasks do not share sampling points, and there is a large number of tasks.* Our experiments, using artificial and real data, validate the approach showing that it can recover the performance of inference with the full sample, and that it performs better than simple baseline sparse approaches as well as the sparse convolved multiple output GP [1].

To summarize, our contribution is threefold. First we propose the first sparse learning algorithm for multi-task GP in the context of the functional grouped mixed-effect model. Second, we develop a variational model selection approach for the proposed sparse model. Finally we evaluate the algorithm and several baseline approaches for multi-task GP, showing that the proposed method performs well against state of the art sparse solutions for other multi-task GP formulations.

2 Nonparametric Bayesian Grouped Mixed-Effect Model

We start by presenting the model which is closely related to the one in [17]. Consider a set of M tasks where the data for the j-th task is given by $\mathcal{D}^j = \{(\boldsymbol{x}_i^j, y_i^j)\}, i = 1, 2, \cdots, N_j$. Given data $\mathcal{D} = \{\mathcal{D}^j\}$, we are interested in learning the nonparametric Bayesian grouped mixed-effect model and using the model to perform inference. The model captures each task f^j as a sum of a mean effect function chosen from a predefined set of K groups and an individual variation (random effect) specific to the j-th task. More precisely,

Assumption 1. *For each j and $\boldsymbol{x} \in \mathcal{X}$, $f^j(\boldsymbol{x}) = \bar{f}_{z_j}(\boldsymbol{x}) + \tilde{f}^j(\boldsymbol{x}), \quad j = 1, \cdots, M$ where $\{\bar{f}_k\}, k = 1, \cdots, K$ and $\tilde{f}^j$ are zero-mean Gaussian processes with*

covariance function $\mathcal{K}_k$ *and* $\widetilde{\mathcal{K}}$*, and* $z_j \in \{1, \cdots, K\}$*. In addition,* $\{\bar{f}_k\}$ *and* $\tilde{f}^j$ *are assumed to be mutually independent.*

The generative process is as follows, where **Dir** and **Multi** denote the Dirichlet and the Multinominal distribution respectively.

1. Draw the processes of the mean effect: $\bar{f}_k(\cdot)|\boldsymbol{\theta}_k \sim \mathcal{GP}(0, \mathcal{K}_k(\cdot,\cdot)), \quad k = 1, 2, \cdots, K$;
2. Draw $\boldsymbol{\pi}|\boldsymbol{\alpha}_0 \sim \mathbf{Dir}(\boldsymbol{\alpha}_0)$;
3. For the j-th task (time series);
   - Draw $z_j|\boldsymbol{\pi} \sim \mathbf{Multi}(\boldsymbol{\pi})$;
   - Draw the random effect: $\tilde{f}^j(\cdot)|\tilde{\boldsymbol{\theta}} \sim \mathcal{GP}(0, \widetilde{\mathcal{K}}(\cdot,\cdot))$;
   - Draw $\boldsymbol{y}^j|z_j, f^j, \boldsymbol{x}^j, \sigma_j^2 \sim \mathcal{N}\left(f^j(\boldsymbol{x}^j), \sigma_j^2 \cdot \mathbb{I}_j\right)$, where $f^j = \bar{f}_{z_j} + \tilde{f}^j$ and where to simplify the notation $\mathbb{I}_j$ stands for $\mathbb{I}_{N_j}$.

When $K = 1$, our model reduces to the classical mixed-effect GP model [7, 5]. Due to the analogy to clustering we sometimes refer to the latent fixed-effect functions as "centers". Let $\breve{\boldsymbol{x}}$ be the concatenation of the examples from all tasks $\breve{\boldsymbol{x}} = (\boldsymbol{x}_i^j)$, and similarly let $\breve{\boldsymbol{y}} = (\boldsymbol{y}_i^j)$, where $i = 1, 2, \cdots, N_j, j = 1, 2, \cdots, M$ and $N = \sum_j N_j$. When the assignment of tasks into groups is known, the likelihood decomposes into separate terms and the predictive distribution can be obtained directly.

$$\begin{aligned} \mathbb{E}(f^j(\boldsymbol{x}^*)|\mathbf{Z}) &= \mathbf{C}^\dagger(\boldsymbol{x}^*, \breve{\boldsymbol{x}})(\mathbf{C}^\dagger(\breve{\boldsymbol{x}}, \breve{\boldsymbol{x}}) + \mathcal{I})^{-1}\breve{\boldsymbol{y}} \\ \mathbf{Cov}(f^j(\boldsymbol{x}^*)|\mathbf{Z}) &= \mathbf{C}^\dagger(\boldsymbol{x}^*, \boldsymbol{x}^*) - \mathbf{C}^\dagger(\boldsymbol{x}^*, \breve{\boldsymbol{x}})(\mathbf{C}^\dagger(\breve{\boldsymbol{x}}, \breve{\boldsymbol{x}}) + \mathcal{I})^{-1}\mathbf{C}^\dagger(\breve{\boldsymbol{x}}, \boldsymbol{x}^*), \end{aligned} \tag{1}$$

where the covariance matrix $\mathbf{C}^\dagger$ is given by $\mathbf{C}^\dagger((\boldsymbol{x}_i^j), (\boldsymbol{x}_k^l)) = \delta_{z_j, z_l}\mathcal{K}_{z_j}(\boldsymbol{x}_i^j, \boldsymbol{x}_k^l) + \delta_{jl} \cdot \widetilde{\mathcal{K}}(\boldsymbol{x}_i^j, \boldsymbol{x}_k^l)$, and $\mathcal{I} = \bigoplus_j \sigma_j^2 \mathbb{I}_j$ ($\bigoplus$ denotes the matrix direct sum). The marginal distribution is $\Pr(\breve{\boldsymbol{y}}|\breve{\boldsymbol{x}}) \sim \mathcal{N}(\mathbf{0}, \mathbf{C}^\dagger(\breve{\boldsymbol{x}}, \breve{\boldsymbol{x}}) + \mathcal{I})$, which can be used for model selection when group membership is known or if our model only allows for one group. This model works well in that sharing the information improves predictive performance but, as the number of tasks grows, the dimension N increases leading to slow inference scaling as $\mathcal{O}(N^3)$. In other words, even though each task may have a very small sample, the multi-task inference problem becomes infeasible when the number of tasks is large. This holds even if we have the group structure.

For single task GP regression, in order to reduce the computational cost, several sparse GP approaches have been proposed [11–13, 15]. In general, these methods approximate the GP with a small number $m \ll N$ of support variables and perform inference using this subset and the corresponding function values $\mathbf{f}_m$. Different approaches differ in how they choose the support variables and the simplest approach is to choose a random subset of the given data points. Recently, Titsias [15] introduced a sparse method based on variational inference using a set $\mathcal{X}_m$ of support variables, which are independent from the training points. In this approach, the support variables $\mathcal{X}_m$ are chosen to maximize a variational lower bound on the marginal likelihood, therefore providing a clear

methodology for the choice of the support set. Later, [2] extended this idea to derive variational approximation for the sparse convolved multiple output GPs.

Developing a sparse solution for our model is significantly more complex than the single task case because of the need to perform inference over multiple tasks, and even more so because the group structure is not known in advance. In this paper, we propose a variational method to solve the learning problem for the mixture model (both full and sparse) as well as choosing the optimal support variables for the sparse model. As in the case of sparse methods for single task GP, we want to introduce a small set of m auxiliary support variables $\mathcal{X}_m$ and base the learning and inference on these points. For the multi-task case, each $\tilde{f}^j(\cdot)$ is specific to the j-th task. Therefore, it makes sense to induce values only for the fixed-effect portion. Our sparse model picks a separate set $\mathcal{X}_m^k$ for each group and uses the fixed-effect portion $\boldsymbol{\eta}_k = \bar{f}_k(\mathcal{X}_m^k)$ for inference. The details of this construction for learning and for prediction are developed in the next two sections.

3 Learning the Sparse Model

In this section we show how to perform the learning via variational approximation. As mentioned above, for the k-th mixed-effect (or center), we introduce m_k auxiliary inducing support variables $\mathcal{X}_m^k$ and the hidden variable $\boldsymbol{\eta}_k = \bar{f}_k(\mathcal{X}_m^k)$, which is the value of k-th fixed-effect function evaluated at $\mathcal{X}_m^k$.

Let $\mathfrak{f}_k = \bar{f}_k(\breve{\boldsymbol{x}}) \in \mathbb{R}^N$ denote the function values of the k-th mean effect so that $\mathbf{f}_k^j = \bar{f}_k(\boldsymbol{x}^j) \in \mathbb{R}^{N_j}$ is the sub-vector of $\mathfrak{f}_k$ corresponding to the j-th task. Let $\tilde{\mathbf{f}}^j = \tilde{f}(\boldsymbol{x}^j) \in \mathbb{R}^{N_j}$ be the values of the random effect at $\boldsymbol{x}^j$. Denote the collection of the hidden variables as $\mathfrak{F} = \{\mathfrak{f}_k\}, \widetilde{\mathcal{F}} = \{\tilde{\mathbf{f}}^j\}, \mathbf{H} = \{\boldsymbol{\eta}_k\}$, $\mathbf{Z} = \{z_j\}$, and $\boldsymbol{\pi}$. In addition let $\mathbf{c}_{*j}^k = \mathcal{K}_k(\boldsymbol{x}^*, \boldsymbol{x}^j)$, $\mathbf{C}_{jj}^k = \mathcal{K}_k(\boldsymbol{x}^j, \boldsymbol{x}^j), \mathbf{C}_{jk} = \mathcal{K}_k(\boldsymbol{x}^j, \mathcal{X}_m^k)$ and $\mathbf{C}_{kk} = \mathcal{K}_k(\mathcal{X}_m^k, \mathcal{X}_m^k)$, and similarly $\widetilde{\mathbf{c}}_{*j} = \widetilde{\mathcal{K}}(\boldsymbol{x}^*, \boldsymbol{x}^j), \widetilde{\mathbf{C}}_{jj} = \widetilde{\mathcal{K}}(\boldsymbol{x}^j, \boldsymbol{x}^j)$ and $\widehat{\mathbf{C}}_{jj} = \widetilde{\mathbf{C}}_{jj} + \sigma_j^2 \mathbb{I}_j$ where $\mathbb{I}_j$ stands for $\mathbb{I}_{N_j}$.

To learn the sparse model we need to maximize the marginal likelihood $\Pr(\breve{\boldsymbol{y}}|\breve{\boldsymbol{x}})$, which cannot be evaluated directly. In the following we develop a variational lower bound for this quantity. To this end, we need the complete data likelihood and the variational distribution. The complete data likelihood is given by

$$\Pr(\breve{\boldsymbol{y}}, \mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}, \mathbf{Z}, \boldsymbol{\pi}) = \Pr(\breve{\boldsymbol{y}}|\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{Z}) \Pr(\mathfrak{F}|\mathbf{H}) \Pr(\mathbf{Z}|\boldsymbol{\pi}) \Pr(\boldsymbol{\pi}) \Pr(\widetilde{\mathcal{F}}) \Pr(\mathbf{H}), \quad (2)$$

$$\Pr(\mathbf{H}) = \prod_{k=1}^{K} \Pr(\boldsymbol{\eta}_k), \ \Pr(\widetilde{\mathcal{F}}) = \prod_{j=1}^{M} \Pr(\tilde{\mathbf{f}}^j), \Pr(\boldsymbol{\pi}) = \mathbf{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_0), \ \Pr(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{j=1}^{M} \prod_{k=1}^{K} \pi_k^{z_{jk}}$$

$$\Pr(\mathfrak{F}|\mathbf{H}) = \prod_{k=1}^{K} \Pr(\mathfrak{f}_k|\boldsymbol{\eta}_k), \ \Pr(\breve{\boldsymbol{y}}|\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{Z}) = \prod_{j=1}^{M} \prod_{k=1}^{K} \left[\Pr(\boldsymbol{y}^j|\tilde{\mathbf{f}}^j, \mathfrak{f}_k)\right]^{z_{jk}}$$

where, as usual, $\{z_{jk}\}$ represent z_j as a unit vector.

Next we approximate the true posterior $\Pr(\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}, \mathbf{Z}, \boldsymbol{\pi}|\breve{\boldsymbol{y}})$ on the hidden variables using the following variational distribution

$$q(\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}, \mathbf{Z}, \boldsymbol{\pi}) = q(\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}|\mathbf{Z})q(\mathbf{Z})q(\boldsymbol{\pi}) \tag{3}$$

where $q(\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}|\mathbf{Z}) = \Pr(\widetilde{\mathcal{F}}|\mathfrak{F}, \mathbf{Z}, \breve{\boldsymbol{y}}) \Pr(\mathfrak{F}|\mathbf{H})\boldsymbol{\Phi}(\mathbf{H})$, which equals

$$\prod_{j=1}^{M}\prod_{k=1}^{K}\left[\Pr(\tilde{\mathbf{f}}^j|\mathfrak{f}_k, \boldsymbol{y}^j)\right]^{z_{jk}} \prod_{k=1}^{K}\Pr(\mathfrak{f}_k|\boldsymbol{\eta}_k)\phi(\boldsymbol{\eta}_k).$$

This generalizes the variational form used by [15] to handle the multiple tasks, their grouping and the individual variations of each task. One can see that the variational distribution is not completely factorized (i.e., some dependencies are preserved) but also not completely in free form in that the value of some of the factors is already determined. In particular, $q(\cdot)$ preserves the exact form of $\Pr(\tilde{\mathbf{f}}^j|\mathfrak{f}_k, \boldsymbol{y}^j)$ and in using $\Pr(\mathfrak{f}_k|\boldsymbol{\eta}_k)$ it preserves some information but implicitly assumes that $\boldsymbol{\eta}_k$ is a sufficient statistic for $\mathfrak{f}_k$. The free form $\phi(\boldsymbol{\eta}_k)$ corresponds to $\Pr(\boldsymbol{\eta}_k|\mathcal{D})$ but allows it to diverge from this value to compensate for the assumption that $\boldsymbol{\eta}_k$ is sufficient. Notice that we are not making any assumption about the sufficiency of $\boldsymbol{\eta}_k$ in the generative model and the approximation is entirely due to the variational distribution. An additional assumption is needed in the next section to derive a simplified form of the predictive distribution.

The variational lower bound, denoted as F_V, is given by:

$$\begin{aligned}
\Pr(\breve{\boldsymbol{y}}|\breve{\boldsymbol{x}}) \geqslant F_V &= \int q(\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}, \mathbf{Z}, \boldsymbol{\pi}) \times \log\left[\frac{\Pr(\breve{\boldsymbol{y}}, \mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}, \mathbf{Z}, \boldsymbol{\pi})}{q(\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}, \mathbf{Z}, \boldsymbol{\pi})}\right] d\mathfrak{F}\, d\widetilde{\mathcal{F}}\, d\mathbf{H}\, d\mathbf{Z}\, d\boldsymbol{\pi} \\
&= \int q(\mathbf{Z})q(\boldsymbol{\pi}) \log\left[\frac{\Pr(\boldsymbol{\pi})\Pr(\mathbf{Z}|\boldsymbol{\pi})}{q(\mathbf{Z})q(\boldsymbol{\pi})}\right] d\boldsymbol{\pi} d\mathbf{Z} \\
&\quad + \int q(\mathbf{Z})q(\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}|\mathbf{Z}) \log\left[\frac{\Pr(\breve{\boldsymbol{y}}|\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{Z})\Pr(\mathfrak{F}|\mathbf{H})\Pr(\widetilde{\mathcal{F}})\Pr(\mathbf{H})}{q(\mathfrak{F}, \widetilde{\mathcal{F}}, \mathbf{H}|\mathbf{Z})}\right] d\mathfrak{F} d\widetilde{\mathcal{F}} d\mathbf{H} d\mathbf{Z}
\end{aligned}$$

After some algebraic manipulation, the variational lower bound can be rewritten as follows.

$$\begin{aligned}
F_V &= \int q(\mathbf{Z})q(\pi) \log\left[\frac{\Pr(\boldsymbol{\pi})\Pr(\mathbf{Z}|\boldsymbol{\pi})}{q(\mathbf{Z})q(\boldsymbol{\pi})}\right] d\boldsymbol{\pi} d\mathbf{Z} \\
&\quad + \int q(\mathbf{Z})\left[\int \prod_{k=1}^{K}\phi(\boldsymbol{\eta}_k)\left\{\log G(\mathbf{Z}, \mathbf{H}, \breve{\boldsymbol{y}}) + \sum_{k=1}^{K}\log\left[\frac{\Pr(\boldsymbol{\eta}_k)}{\phi(\boldsymbol{\eta}_k)}\right]\right\} d\mathbf{H}\right] d\mathbf{Z}
\end{aligned} \tag{4}$$

where $\log G(\mathbf{Z}, \mathbf{H}, \breve{\boldsymbol{y}})$ equals

$$\int \Pr(\widetilde{\mathcal{F}}|\mathfrak{F}, \mathbf{Z}, \breve{\boldsymbol{y}}) \Pr(\mathfrak{F}|\mathbf{H}) \log\left[\prod_{j=1}^{M}\prod_{k=1}^{K}\left[\frac{\Pr(\boldsymbol{y}^j|\mathfrak{f}_k, \tilde{\mathbf{f}}^j)\Pr(\tilde{\mathbf{f}}^j)}{\Pr(\tilde{\mathbf{f}}^j|\mathfrak{f}_k, \boldsymbol{y}^j)}\right]^{z_{jk}}\right] d\mathfrak{F} d\widetilde{\mathcal{F}}.$$

In Section 3.1, we show that $\log G(\mathbf{Z}, \mathbf{H}, \breve{\boldsymbol{y}})$ can be decomposed as $\log G(\mathbf{Z}, \mathbf{H}, \breve{\boldsymbol{y}}) = \sum_{j=1}^{M} \sum_{k=1}^{K} z_{jk} \log G(\boldsymbol{\eta}_k, \boldsymbol{y}^j)$, where

$$\log G(\boldsymbol{\eta}_k, \boldsymbol{y}^j) = \log \left[\mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, \widehat{\mathbf{C}}_{jj}) \right] - \frac{1}{2} \mathbf{Tr} \left[(\mathbf{C}_{jj}^k - \boldsymbol{Q}_{jj}^k) \widehat{\mathbf{C}}_{jj}^{-1} \right], \tag{5}$$

where $\boldsymbol{\alpha}_j^k = \mathbf{C}_{jk} \mathbf{C}_{kk}^{-1} \boldsymbol{\eta}_k$ and $\boldsymbol{Q}_{jj}^k = \mathbf{C}_{jk} \mathbf{C}_{kk}^{-1} \mathbf{C}_{kj}$.

To optimize the parameters we use the variational EM algorithm. In the **Variational E-Step**, we estimate $q^*(\mathbf{Z}), q^*(\boldsymbol{\pi})$ and $\{\phi^*(\boldsymbol{\eta}_k)\}$.

To get the variational distribution $q^*(\mathbf{Z})$, we take derivative of F_V w.r.t. $q(\mathbf{Z})$ and set it to 0. Solving for $q(\mathbf{Z})$, we get

$$q^*(\mathbf{Z}) = \prod_{j=1}^{M} \prod_{k=1}^{K} r_{jk}^{z_{jk}}, \qquad r_{jk} = \frac{\rho_{jk}}{\sum_{k=1}^{K} \rho_{jk}}$$
$$\log \rho_{jk} = \mathbb{E}_{q(\boldsymbol{\pi})}[\log \pi_k] + \mathbb{E}_{\phi(\boldsymbol{\eta}_k)}[\log G(\boldsymbol{\eta}_k, \boldsymbol{y}^j)],$$

where $\mathbb{E}_{q(\boldsymbol{\pi})}[\log \pi_k] = \Psi(\alpha_k) - \Psi(\sum_k \alpha_k)$ where Ψ is the digamma function, α_k is defined below, and $\mathbb{E}_{\phi(\boldsymbol{\eta}_k)}[\log G(\boldsymbol{\eta}_k, \boldsymbol{y}^j)]$ is given below in (16).

Similarly, $q^*(\boldsymbol{\pi})$ can be obtained as $q^*(\boldsymbol{\pi}) = \mathbf{Dir}(\boldsymbol{\pi} | \boldsymbol{\alpha})$ where $\alpha_k = \alpha_0 + N_k$ and $N_k = \sum_{j=1}^{K} r_{jk}$.

The final step is to get the variational distribution of $\phi^*(\boldsymbol{\eta}_k), k = 1, \cdots, K$. Notice that only the second term of F_V is a function of $\phi(\boldsymbol{\eta}_k)$ and it can be rewritten as

$$\sum_{k=1}^{K} \int \phi(\boldsymbol{\eta}_k) \left\{ \left[\sum_{j=1}^{M} \mathbb{E}_{q(\mathbf{Z})}[z_{jk}] \log G(\boldsymbol{\eta}_k, \boldsymbol{y}^j) \right] + \log \left[\frac{\Pr(\boldsymbol{\eta}_k)}{\phi(\boldsymbol{\eta}_k)} \right] \right\} d\boldsymbol{\eta}_k. \tag{6}$$

Thus, our task reduces to find each $\phi^*(\boldsymbol{\eta}_k)$ separately. Taking the derivative of (6) w.r.t. $\phi(\boldsymbol{\eta}_k)$ and setting it to 0, we obtain

$$\phi^*(\boldsymbol{\eta}_k) \propto \prod_{j=1}^{M} \left[\mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, \widehat{\mathbf{C}}_{jj}) \right]^{\mathbb{E}_{q(\mathbf{Z})}[z_{jk}]} \Pr(\boldsymbol{\eta}_k). \tag{7}$$

Thus, we have

$$\phi^*(\boldsymbol{\eta}_k) \propto \exp \left\{ -\frac{1}{2} \boldsymbol{\eta}_k^T \left(\mathbf{C}_{kk}^{-1} \boldsymbol{\Phi} \mathbf{C}_{kk}^{-1} \right) \boldsymbol{\eta}_k + \boldsymbol{\eta}_k^T \left(\mathbf{C}_{kk}^{-1} \sum_{j=1}^{M} r_{jk} \mathbf{C}_{kj} [\widehat{\mathbf{C}}_{jj}]^{-1} \boldsymbol{y}_j \right) \right\},$$

where $\boldsymbol{\Phi} = \mathbf{C}_{kk} + \sum_{j=1}^{M} r_{jk} \mathbf{C}_{kj} [\widehat{\mathbf{C}}_{jj}]^{-1} \mathbf{C}_{jk}$. Completing the square yields the Gaussian distribution $\phi^*(\boldsymbol{\eta}_k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where

$$\boldsymbol{\mu}_k = \mathbf{C}_{kk} \boldsymbol{\Phi}^{-1} \sum_{j=1}^{M} r_{jk} \mathbf{C}_{kj} [\widehat{\mathbf{C}}_{jj}]^{-1} \boldsymbol{y}_j, \quad \boldsymbol{\Sigma}_k = \mathbf{C}_{kk} \boldsymbol{\Phi}^{-1} \mathbf{C}_{kk}. \tag{8}$$

In the **Variational M-Step**, based on the previous estimated variational distribution, we wish to find hyperparameters that maximize the variational lower bound F_V. The terms that depend on the hyperparameters $\mathbf{\Theta}$ and the inducing variables $\mathcal{X}_m = \{\mathcal{X}_m^k\}$ are given in (6). Therefore, using (5) again, we can express $F_V(\mathcal{X}_m, \mathbf{\Theta})$ as

$$\sum_{k=1}^{K} \mathbb{E}_{\phi^*(\boldsymbol{\eta}_k)} \left\{ \log \left[\frac{\prod_j \left[\mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, \widehat{\mathbf{C}}_{jj}) \right]^{r_{jk}} \Pr(\boldsymbol{\eta}_k)}{\phi^*(\boldsymbol{\eta}_k)} \right] \right\} - \frac{1}{2} \sum_{k,j}^{m} r_{jk} \mathbf{Tr} \left[(\mathbf{C}_{jj}^k - \boldsymbol{Q}_{jj}) \widehat{\mathbf{C}}_{jj}^{-1} \right].$$

From (7), we know that the term inside the log is constant, and therefore, extracting the log from the integral and cancelling the $\phi^*(\boldsymbol{\eta}_k)$ terms we see that the k'th element of first term is equal to the logarithm of

$$\int \prod_{j=1}^{M} \left[\mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, \widehat{\mathbf{C}}_{jj}) \right]^{r_{jk}} \Pr(\boldsymbol{\eta}_k) d\boldsymbol{\eta}_k. \tag{9}$$

We next show how this multivariate integral can be evaluated. First we can write $\left[\mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, \widehat{\mathbf{C}}_{jj}) \right]^{r_{jk}} = A_{jk} \mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, r_{jk}^{-1} \widehat{\mathbf{C}}_{jj})$, where $A_{jk} = (r_{jk})^{\frac{N_j}{2}} (2\pi)^{\frac{N_j(1-r_{jk})}{2}} |\widehat{\mathbf{C}}_{jj}|^{\frac{1-r_{jk}}{2}}$. Thus, we have $\prod_j \left[\mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, \widehat{\mathbf{C}}_{jj}) \right]^{r_{jk}} = \left[\prod_j A_{jk} \right] \prod_j \mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, r_{jk}^{-1} \widehat{\mathbf{C}}_{jj})$. As the first part is not a function of $\boldsymbol{\eta}_k$, for the integration we are only interested in the second part. Since $\breve{\boldsymbol{y}}$ is the concatenation of all $\boldsymbol{y}^j$'s, we can write

$$\prod_{j=1}^{M} \mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, r_{jk}^{-1} \widehat{\mathbf{C}}_{jj}) = \mathcal{N}(\breve{\boldsymbol{y}} | \boldsymbol{\Lambda}_k \mathbf{C}_{kk}^{-1} \boldsymbol{\eta}_k, \widehat{\mathbf{C}}^k), \tag{10}$$

where $\boldsymbol{\Lambda}_k = [\mathbf{C}_{1k}^T, \mathbf{C}_{2k}^T, \cdots, \mathbf{C}_{Mk}^T]^T \in \mathbb{R}^{N, m_k}$ and $\widehat{\mathbf{C}}^k = \bigoplus_{j=1}^{M} r_{jk}^{-1} \widehat{\mathbf{C}}_{jj}^k \in \mathbb{R}^{N,N}$, which is the block diagonal matrix with element $r_{jk}^{-1} \widehat{\mathbf{C}}_{jj}^k$. Therefore, the integral can the written as the following marginal distribution of $\Pr(\breve{\boldsymbol{y}}|k)$,

$$\int \prod_{j=1}^{M} \mathcal{N}(\boldsymbol{y}^j | \boldsymbol{\alpha}_j^k, r_{jk}^{-1} \widehat{\mathbf{C}}_{jj}) \Pr(\boldsymbol{\eta}_k) d\boldsymbol{\eta}_k = \int \mathcal{N}(\breve{\boldsymbol{y}} | \boldsymbol{\Lambda}_k \mathbf{C}_{kk}^{-1} \boldsymbol{\eta}_k, \widehat{\mathbf{C}}^k) \Pr(\boldsymbol{\eta}_k) d\boldsymbol{\eta}_k. \tag{11}$$

Using the fact that $\Pr(\boldsymbol{\eta}_k) = \mathcal{N}(\mathbf{0}, \mathbf{C}_{kk})$ and observing that (10) is a conditional Gaussian, we have $\Pr(\breve{\boldsymbol{y}}|k) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_k \mathbf{C}_{kk}^{-1} \boldsymbol{\Lambda}_k^T + \widehat{\mathbf{C}}^k)$. Using this form and the portion of A_{jk} that depends on the parameters we get the variational lower bound $F_V(\mathcal{X}_m, \mathbf{\Theta})$, which equals

$$\sum_{k=1}^{K} \log \Pr(\breve{\boldsymbol{y}}|k) + \frac{K-1}{2} \sum_{j=1}^{M} \log |\widehat{\mathbf{C}}_{jj}| - \frac{1}{2} \sum_{j,k} r_{jk} \mathbf{Tr} \left[(\mathbf{C}_{jj}^k - \boldsymbol{Q}_{jj}) \widehat{\mathbf{C}}_{jj}^{-1} \right]. \tag{12}$$

Notice that when the number of tasks and the number of centers are both 1, we recover the results in [15] provided that the random effect is independent white noise.

Using the ideas in the previous derivation, the direct inference for the full model can also be obtained where $\boldsymbol{\eta}_k$ is substituted with $\mathfrak{f}_k$ and the variational lower bound becomes

$$F_V(\mathcal{X}_m, \boldsymbol{\Theta}) = \sum_{k=1}^{K} \log \mathcal{N}(\breve{\boldsymbol{y}}|\mathbf{0}, \mathbf{C}_{kk} + \widehat{\mathbf{C}}^k) + \frac{K-1}{2} \sum_{j=1}^{M} \log |\widehat{\mathbf{C}}_{jj}|. \tag{13}$$

We have explicitly written the parameters that can be chosen to further optimize the lower bound (12), namely the support inputs $\{\mathcal{X}_m^k\}$, and the set of hyper-parameters $\boldsymbol{\Theta}$ which is composed of $\{\boldsymbol{\theta}_k\}$ and $\{\tilde{\boldsymbol{\theta}}\}$ in $\mathcal{K}_k$ and $\widetilde{\mathcal{K}}$ respectively.

By calculating derivatives of (12) we can optimize the lower bound using a gradient based method. This can be done by making use of the special form of the covariance matrix $\boldsymbol{\Lambda}_k \mathbf{C}_{kk}^{-1} \boldsymbol{\Lambda}_k^T + \widehat{\mathbf{C}}^k$, the matrix inversion formula, the chain rule for derivatives, and sequencing the matrix operations appropriately (details omitted due to space constraints). The complexity of evaluating the derivative of (12) is $\mathcal{O}(N \sum_k m_k^2 + \sum_k m_k^3 + \sum_j N_j^3)$. In our implementation, we use stochastic coordinate descent, where at each iteration, one coordinate (parameter) is chosen at random and we perform gradient descent on that coordinate.

3.1 Evaluating $\log G(\mathbf{Z}, \mathbf{H}, \breve{\boldsymbol{y}})$

In this section, we wish to evaluate $\log G(\mathbf{Z}, \mathbf{H}, \breve{\boldsymbol{y}})$, which equals

$$\begin{aligned}
&\int \prod_{l=1}^{M} \prod_{p=1}^{K} \left[\Pr(\tilde{\mathbf{f}}^l|\mathfrak{f}^p, \boldsymbol{y}^l)\right]^{z_{lp}} \prod_{v=1}^{K} \Pr(\mathfrak{f}_v|\boldsymbol{\eta}_v) \times \sum_{j=1}^{M} \sum_{k=1}^{K} z_{jk} \log \left[\frac{\Pr(\boldsymbol{y}^j|\mathfrak{f}_k, \tilde{\mathbf{f}}^j) \Pr(\tilde{\mathbf{f}}^j)}{\Pr(\tilde{\mathbf{f}}^j|\mathfrak{f}_k, \boldsymbol{y}^j)}\right] d\mathfrak{F} d\widetilde{\mathcal{F}} \\
&= \sum_{j=1}^{M} \sum_{k=1}^{K} z_{jk} \left[\int \Pr(\tilde{\mathbf{f}}^j|\mathfrak{f}_k, \boldsymbol{y}^j) \Pr(\mathfrak{f}_k|\boldsymbol{\eta}_k) \times \log \left[\frac{\Pr(\boldsymbol{y}^j|\mathfrak{f}_k, \tilde{\mathbf{f}}^j) \Pr(\tilde{\mathbf{f}}^j)}{\Pr(\tilde{\mathbf{f}}^j|\mathfrak{f}_k, \boldsymbol{y}^j)}\right] d\mathfrak{f}_k d\tilde{\mathbf{f}}^j\right],
\end{aligned} \tag{14}$$

where the second line holds because in the sum indexed by l and p all the product measures $\prod_{l \neq j, p \neq k} \left[\Pr(\tilde{\mathbf{f}}^l|\mathfrak{f}^p, \boldsymbol{y}^l)\right]^{z_{lp}}$ are integrated to 1, leaving only the $\Pr(\tilde{\mathbf{f}}^j|\mathfrak{f}^k, \boldsymbol{y}^j)$. Denote the term inside the brackets by $\log G(\boldsymbol{\eta}_k, \boldsymbol{y}^j)$; this term can be evaluated as

$$\begin{aligned}
&\int \Pr(\tilde{\mathbf{f}}^j|\mathfrak{f}_k, \boldsymbol{y}^j) \Pr(\mathfrak{f}_k|\boldsymbol{\eta}_k) \times \log \left[\Pr(\boldsymbol{y}^j|\mathfrak{f}_k, \tilde{\mathbf{f}}^j) \Pr(\tilde{\mathbf{f}}^j) \cdot \frac{\Pr(\boldsymbol{y}^j|\mathfrak{f}_k)}{\Pr(\boldsymbol{y}^j|\mathfrak{f}_k, \tilde{\mathbf{f}}^j) \Pr(\tilde{\mathbf{f}}^j|\mathfrak{f}_k)}\right] d\mathfrak{f}_k d\tilde{\mathbf{f}}^j \\
&= \int \Pr(\mathfrak{f}_k|\boldsymbol{\eta}_k) \log \left[\Pr(\boldsymbol{y}^j|\mathfrak{f}_k)\right] d\mathfrak{f}_k = \int \Pr(\mathbf{f}_k^j|\boldsymbol{\eta}_k) \log \left[\Pr(\boldsymbol{y}^j|\mathbf{f}_k^j)\right] d\mathbf{f}_k^j
\end{aligned} \tag{15}$$

where the last line holds because of the independence between $\tilde{\mathbf{f}}^j$ and $\mathfrak{f}_k$. Noticing that evaluating (15) involves marginalization over Gaussians, after some algebraic manipulation, we obtain (5).

Furthermore, marginalizing out $\boldsymbol{\eta}_k$, we get that $\mathbb{E}_{\phi^*(\boldsymbol{\eta}_k)} \log G(\boldsymbol{\eta}_k, \boldsymbol{y}^j)$ equals

$$\log \left[\mathcal{N}(\boldsymbol{y}^j|\boldsymbol{\mu}_k, \widehat{\mathbf{C}}_{jj})\right] - \frac{1}{2} \mathbf{Tr} \left[\mathbf{C}_{jk} \mathbf{C}_{kk}^{-1} (\boldsymbol{\Sigma}_k - \mathbf{C}_{kk}) \mathbf{C}_{kk}^{-1} \mathbf{C}_{jk} \widehat{\mathbf{C}}_{jj}^{-1}\right]. \tag{16}$$

4 Prediction Using the Sparse Model

The proposed sparse model can be used for two types of problems. Prediction for existing tasks and prediction for a newly added task. We start with deriving the predictive distribution for existing tasks. Given any task j, our goal is to calculate the predictive distribution $\Pr(f^j(\boldsymbol{x}^*)|\mathcal{D})$ at new input point $\boldsymbol{x}^*$, which can be written as

$$\sum_{k=1}^{K} \Pr(f^j(\boldsymbol{x}^*)|z_{jk}=1,\mathcal{D}) \Pr(z_{jk}=1|\mathcal{D}) = \sum_{k=1}^{K} r_{jk} \Pr(f^j(\boldsymbol{x}^*)|z_{jk}=1,\mathcal{D}). \tag{17}$$

That is, because z_{jk} form a partition we can focus on calculating $\Pr(f^j(\boldsymbol{x}^*)|z_{jk}=1,\mathcal{D})$ and then combine the results using the partial labels. Instead of calculating the full Bayesian prediction, one can use *Maximum A Posteriori* (MAP) by assigning the j-th task to the center c such that $c = \operatorname{argmax} \Pr(z_{jk}=1|\mathcal{D})$. Preliminary experiments (not shown here) show that the full Bayesian approach gives better performance. In the following, we will show how to calculate $\Pr(f^j(\boldsymbol{x}^*)|z_{jk}=1,\mathcal{D})$, i.e. the predictive distribution when $f^j = \bar{f}_k + \tilde{f}_j$.

As described before, the full inference is expensive and therefore we wish to use the variational approximation for the prediction as well. The key idea is that $\boldsymbol{\eta}_k$ contains as much information as $\mathcal{D}$ in terms of making prediction for $\bar{f}_k$. To start with, for each k, it is easy to see that the predictive distribution is Gaussian (conditioned on $z_{jk}=1$) and that it satisfies

$$\begin{aligned}
\mathbb{E}[f^j(\boldsymbol{x}^*)|\mathcal{D}] &= \mathbb{E}[\bar{f}_k(\boldsymbol{x}^*)|\mathcal{D}] + \mathbb{E}[\tilde{f}^j(\boldsymbol{x}^*)|\mathcal{D}] \\
\mathbf{Var}[f^j(\boldsymbol{x}^*)|\mathcal{D}] &= \mathbf{Var}[\bar{f}_k(\boldsymbol{x}^*)|\mathcal{D}] + \mathbf{Var}[\tilde{f}^j(\boldsymbol{x}^*)|\mathcal{D}] + 2\mathbf{Cov}[\bar{f}_k(\boldsymbol{x}^*)\tilde{f}^j(\boldsymbol{x}^*)|\mathcal{D}].
\end{aligned} \tag{18}$$

The above equation is more complex than the predictive distribution for single-task sparse GP because of the coupling induced by $\bar{f}_k(\boldsymbol{x}^*)\tilde{f}^j(\boldsymbol{x}^*)|\mathcal{D}$. We next show how this can be calculated via conditioning.

The calculation of the terms in (18) consists of three parts, i.e. $\Pr(\bar{f}_k(\boldsymbol{x}^*)|\mathcal{D})$, $\Pr(\tilde{f}(\boldsymbol{x}^*)|\mathcal{D})$ and $\mathbf{Cov}[\bar{f}_k(\boldsymbol{x}^*)\tilde{f}_j(\boldsymbol{x}^*)|\mathcal{D}]$. Using the approximation of the variational form given in (3), we have the following facts:

1. $\boldsymbol{\eta}_k|\mathcal{D} \sim \phi^*(\boldsymbol{\eta}_k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are given in (8).

2. $\boldsymbol{\eta}_k$ is sufficient for $\mathfrak{f}_k$, i.e. $\Pr(\mathfrak{f}_k|\boldsymbol{\eta}_k,\mathcal{D}) = \Pr(\mathfrak{f}_k|\boldsymbol{\eta}_k)$. Since we are interested in prediction for each task separately, by marginalizing out the tasks other than j, we also have $\Pr(\mathbf{f}_k^j|\boldsymbol{\eta}_k,\mathcal{D}) = \Pr(\mathbf{f}_k^j|\boldsymbol{\eta}_k)$ and

$$\mathbf{f}_k^j|\boldsymbol{\eta}_k,\mathcal{D} \sim \mathcal{N}\left(\mathbf{C}_{jk}\mathbf{C}_{kk}^{-1}\boldsymbol{\eta}_k, \mathbf{C}_{jj}^k - \mathbf{C}_{jk}\mathbf{C}_{kk}^{-1}\mathbf{C}_{kj}\right). \tag{19}$$

3. For $\tilde{f}^j(\boldsymbol{x}^*)$ we can view $\boldsymbol{y}^j - \mathbf{f}_k^j$ as noisy realizations from the same GP as $\tilde{f}^j(\boldsymbol{x}^j)$.

$$\tilde{f}^j(\boldsymbol{x}^*)|\mathbf{f}_k^j,\mathcal{D} \sim \mathcal{N}\left(\widetilde{\mathbf{c}}_{*j}\left[\widetilde{\mathbf{C}}_{jj} + \sigma_j^2\mathbb{I}_j\right]^{-1}(\boldsymbol{y}^j - \mathbf{f}_k^j), \widetilde{\mathbf{c}}_{**} - \widetilde{\mathbf{c}}_{*j}\left[\widetilde{\mathbf{C}}_{jj} + \sigma_j^2\mathbb{I}_j\right]^{-1}\widetilde{\mathbf{c}}_{j*}\right). \tag{20}$$

In order to obtain a sparse form of the predictive distribution we need to make an additional assumption beyond the variational approximation used for training the model. Specifically, we assume that $\boldsymbol{\eta}_k$ is sufficient for $\bar{f}_k(\boldsymbol{x}^*)$, i.e., $\Pr(\bar{f}_k(\boldsymbol{x}^*)|\boldsymbol{\eta}_k, \mathcal{D}) = \Pr(\bar{f}_k(\boldsymbol{x}^*)|\boldsymbol{\eta}_k)$, implying that

$$\bar{f}(\boldsymbol{x}^*)|\boldsymbol{\eta}_k, \mathcal{D} \sim \mathcal{N}\left(\mathbf{c}^k_{*k}\mathbf{C}^{-1}_{kk}\boldsymbol{\eta}_k, \mathbf{c}^k_{**} - \mathbf{c}_{*k}\mathbf{C}^{-1}_{kk}\mathbf{c}_{k*}\right). \tag{21}$$

The above set of conditional distributions also imply that $\bar{f}_k(\boldsymbol{x}^*)$ and $\tilde{f}^j(\boldsymbol{x}^*)$ are independent given $\boldsymbol{\eta}_k$ and $\mathcal{D}$. Next, we can easily get $\Pr(\bar{f}_k(\boldsymbol{x}^*)|\mathcal{D})$ by marginalizing out $\boldsymbol{\eta}_k|\mathcal{D}$ in (21).

Similarly, we can obtain $\Pr(\tilde{f}(\boldsymbol{x}^*)|\mathcal{D})$ by first calculating $\Pr(\mathbf{f}^j_k|\mathcal{D})$ by marginalizing out $\boldsymbol{\eta}_k|\mathcal{D}$ in (19) and then marginalizing out $\mathbf{f}^j_k|\mathcal{D}$ in (20). Finally, for the remaining term we have $\mathbf{Cov}[\bar{f}_k(\boldsymbol{x}^*)\tilde{f}^j(\boldsymbol{x}^*)|\mathcal{D}] = \mathbb{E}\left[\bar{f}_k(\boldsymbol{x}^*)\tilde{f}^j(\boldsymbol{x}^*)|\mathcal{D}\right] - \mathbb{E}[\bar{f}_k(\boldsymbol{x}^*)|\mathcal{D}]\mathbb{E}[\tilde{f}^j(\boldsymbol{x}^*)|\mathcal{D}]$ where

$$\begin{aligned}\mathbb{E}\left[\bar{f}_k(\boldsymbol{x}^*)\cdot\tilde{f}^j(\boldsymbol{x}^*)|\mathcal{D}\right] &= \mathbb{E}_{\boldsymbol{\eta}_k|\mathcal{D}}\mathbb{E}\left[\bar{f}_k(\boldsymbol{x}^*)\cdot\tilde{f}^j(\boldsymbol{x}^*)|\boldsymbol{\eta}_k, \mathcal{D}\right] \\ &= \mathbb{E}_{\boldsymbol{\eta}_k|\mathcal{D}}\left[\mathbb{E}\left[\bar{f}_k(\boldsymbol{x}^*)|\boldsymbol{\eta}_k\right]\cdot\mathbb{E}[\tilde{f}^j(\boldsymbol{x}^*)|\boldsymbol{\eta}_k, \boldsymbol{y}^j]\right]\end{aligned} \tag{22}$$

where the second line holds because, as observed above, the terms are conditionally independent. The first term $\mathbb{E}\left[\bar{f}^j(\boldsymbol{x}^*)|\boldsymbol{\eta}_k\right]$ can be obtained directly from (21). By marginalizing out $\mathbf{f}^j_k|\boldsymbol{\eta}_k$ in (20) we can get the second term. Finally taking expectation w.r.t. $\phi^*(\boldsymbol{\eta}_k|\mathcal{D})$ can be calculated via properties of the multivariate normal distribution.

We have therefore shown how to calculate the predictive distribution in (18). The complexity of these computations is $\mathcal{O}(K(N_j^3 + m^3))$ which is a significant improvement over $\mathcal{O}(KN^3)$ where $N = \sum_j N_j$.

Our model is also useful for making prediction for newly added tasks. Suppose we are given $\{\boldsymbol{x}^{M+1}, \boldsymbol{y}^{M+1}\}$ and we are interested in predicting $f^{M+1}(\boldsymbol{x}^*)$. We use the variational procedure to estimate its partial labels w.r.t. different centers $\Pr(z_{M+1,k} = 1|\mathcal{D})$ and then (17) can be applied for making the prediction. In the variational procedure we update the parameters for Z_{M+1} but keep all other parameters fixed. Since each task has small number of samples, we expect this step to be computationally cheap.

5 Related Work

Our work is related to [15] particularly in terms of the form of the variational distribution of the inducing variables. However, our model is much more complex than the basic GP regression model. With the mixture model and an additional random effect per task, we must take into account the coupling of the random effect and group specific fixed-effect functions. The technical difficulty that the coupling introduces is addressed in our paper, yielding a generalization that is consistent with single-task solution.

The other related thread comes from the area of GP for multi-task learning. Bonilla et al. proposed a model that learns a shared covariance matrix on features and a covariance matrix for tasks that explicitly models the dependency between tasks [4]. They also presented techniques to speed up the inference by using Nystrom approximation of the kernel matrix and incomplete Cholesky decomposition of the task correlations matrix. Their model, which is known as the linear coregionalization model (LCM) is subsumed by the framework of convolved multiple output Gaussian process [1]. The work of [1] also derives sparse solutions which are extensions of different single task sparse GP [13, 9]. Our work differs from the above models in that we allow a random effect for each individual task. As we show in the experimental section, this is important in modeling various applications. If the random effect is replaced with independent white noise, then our model is similar to LCM. To see this, from (17), we recognize that the posterior GP is a convex combination of K independent GPs (mean effect). However, our model is capable of prediction for newly added tasks while the models in [4] and [1] cannot. Further, the proposed model can naturally handle *heterotopic* inputs, where different tasks do not necessarily share the same inputs. In [4], each task is required to have same number of samples so that one can use the property of Kronecker product to derive the EM algorithm.

6 Experimental Evaluation

Our implementation of the algorithm makes use of the gpml package [10] and extends it to implement the required functions. For performance criteria we use the standardized mean square error (SMSE) and the mean standardized log loss (MSLL) that are defined in [11]. We compare the following methods. The first four methods use the same variational inference as described in Section 3. They differ in the form of the variational lower bound they choose to optimize.

1. **Direct Inference**: use full samples as the support variables and optimize (13). When $K = 1$, the marginal likelihood is described in Section 2 and the predictive distribution is (1).
2. **Variational Sparse GP for MTL (MT-VAR)**: the proposed approach.
3. **MTL Subset of Datapoints (MT-SD)**: a subset $\mathcal{X}_m^k$ of size m_k is chosen uniformly from the input points from all tasks $\breve{\boldsymbol{x}}$ for each center. The hyper-parameters are selected using $\mathcal{X}_m^k$ (the support variables are fixed in advance) and their corresponding observations by maximizing the variational lower bound. We call this MT-SD as a multi-task version of SD [11], because in the single center case we can use the marginal likelihood and (1) where the subset $\mathcal{X}_m, \mathcal{Y}_m$ and $\boldsymbol{x}^j, \boldsymbol{y}^j$ serve as the full sample (thus discarding other samples).
4. **MTL Projected Process Approximation (MT-PP)**: the variational lower bound of MT-PP is given by the first two terms of (12) ignoring the trace term, and therefore the optimization chooses different support variables and hyper-parameters. We call it MT-PP because in the single center case it corresponds to a multi-task version of PP [11].

5. **Convolved Multiple Output GP (MGP-FITC, MGP-PITC)**: the approaches proposed in [1]. For all experiments, we use code from [1] with the following setting. The kernel type is set to be `gg`. The hyperparameters parameters and the position of inducing variables are obtained via optimizing the marginal likelihood using a scaled conjugated gradient algorithm. The support variables are initialized as equally spaced points over the range of the inputs. We set the $R_q = 1$, which means that the latent functions share the same covariance function. Whenever possible, we set Q which, roughly speaking, corresponds to the number of centers in our approach, to agree with the number of centers. The number of maximum iterations allowed in the optimization procedure is set to be 200. The number of support variables is controlled in the experiments as in our methods.

Three datasets are used to demonstrate the empirical performance of the proposed approach. The first synthetic dataset contains data sampled according to our model. The second dataset is also synthetic but it is generated from differential equations describing glucose concentration in biological experiments, a problem that has been previously used to evaluate multi-task GP [7]. Finally, we apply the proposed method on a real astrophysics dataset. For all experiments, the kernels for different centers are assumed to be the same. The hyperparameter for the Dirichlet distribution is set to be $\alpha_0 = 1/K$. The inducing variables are initialized to be equally spaced points over the range of the inputs. To initialize, tasks are randomly assigned into groups. We run the conjugate gradient algorithm (`minimize.m`) on a small subset of tasks (100 tasks each having 5 samples) to get the starting values of hyperparameters of the $\widetilde{\mathcal{K}}$ and $\mathcal{K}$, and then follow with the full optimization as above. Finally, we repeat the entire procedure 5 times and choose the one that achieves best variational lower bound. The maximum number of iterations for the stochastic coordinate descent is set to be 50 and the maximum number of iterations for the variational inference is set to be 30. The entire experiment is repeated 10 times to obtain the average performance and error bars.

6.1 Synthetic Data

In the first experiment, we demonstrate the performance of our algorithm on a regression task with artificial data. More precisely, we generated 1000 single-center tasks where each $f^j(x) = \bar{f}(x) + \tilde{f}^j(x)$ is generated on the interval $x \in [-10, 10]$. Each task has 5 samples. The fixed-effect function is sampled from a GP with covariance function $\mathbf{Cov}[\bar{f}(t_1), \bar{f}(t_2)] = e^{-(t_1-t_2)^2/2}$. The individual effect $\tilde{f}^j$ is sampled via a GP with the covariance function $\mathbf{Cov}[\tilde{f}^j(t_1), \tilde{f}^j(t_2)] = 0.25e^{-(t_1-t_2)^2/2}$. The noise level σ^2 is set to be 0.1. The sample points $\boldsymbol{x}^j$ for each task are sampled uniformly in the interval $[-10, 10]$ and the 100 test samples are chosen equally spaced in the same interval. The fixed-effect curve is generated by drawing a single realization from the distribution of $\bar{\mathbf{f}}$ while the $\{\mathbf{f}^j\}$ are sampled i.i.d. from their common prior. We set the number of latent functions $Q = 1$ for MGP. The results are shown in Fig. (1). The top row shows qualitative results

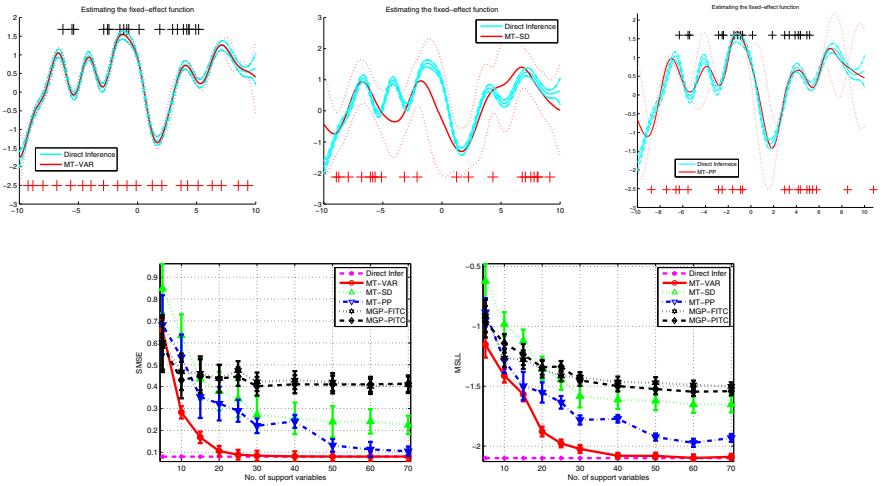


Fig. 1. Synthetic Data. Top row: Predictive distribution for the fixed-effect. The solid line denotes the predictive mean and the corresponding dotted line is the predictive variance. The black crosses (at the top) are the initial value of the support variables and the red ones (at the bottom) are their values after learning process. Bottom row: The average SMSE and MSLL for all the tasks.

for one run using 20 support variables. We restrict the initial support variables to be in $[-7, 7]$ on purpose to show that the proposed method is capable of finding the optimal inducing variables. It is clear that the predictive distribution of the proposed method is much closer to the results of direct inference. The bottom row gives quantitative results for SMSE and MSLL showing the same, as well as showing that with 40 pseudo inputs the proposed method recovers the performance of full inference. The MGP performs poorly on this dataset, indicating that it is not sufficient to capture the random effect. We also see a large computational advantage over MGP in this experiment. When the number of inducing variables is 20, the training time for FITC (the time for constructing the sparse model plus the time for optimization) is 1515.19 sec. while the proposed approach is about 7 times faster (201.81 sec.).

6.2 Simulated Glucose Data

We evaluate our method to reconstruct the glucose profiles in an intravenous glucose tolerance test (IVGTT) [16, 7] where [7] developed an online multi-task GP solution for the case where sample points are frequently shared among tasks. This provides a more realistic test of our algorithm because data is not generated explicitly by our model. We follow previous work and generate the data using

minimal models of glucose which is commonly used to analyze glucose and insulin IVGTT data [16], as follows

$$\begin{aligned} \dot{G}(t) &= -[S_G + X(t)]G(t) + S_G \cdot G_b + \delta(t) \cdot D/V \\ \dot{X}(t) &= -p_2 \cdot X(t) + p_2 \cdot S_I \cdot [I(t) - I_b] \\ G(0) &= G_b, \quad X(0) = 0 \end{aligned} \tag{23}$$

where D denotes the glucose dose, $G(t)$ is plasma glucose concentration and $I(t)$ is the plasma insulin concentration which is assumed to be known. G_b and I_b are the glucose and insulin base values. $X(t)$ is the insulin action and $\delta(t)$ is the Dirac delta function. S_G, S_I, p_2, V are four parameters of this model.

We generate 1000 synthetic subjects (tasks) following the setup in previous work: 1) the four parameters are sampled from a multivariate Gaussian with the results from the normal group in Table 1 of [16], 2) $I(t)$ is obtained via spline interpolation using the real data in [16]; 3) G_b is fixed to be 84 and D is set to be 300; 4) $\delta(t)$ is simulated using a Gaussian profile with support on the positive axis and the standard deviation (SD) randomly drawn from a uniform distribution on the interval $[0, 1]$; 5) Noise is added to the observations with $\sigma^2 = 1$. Each task has 5 measurements chosen uniformly from the interval $[1, 240]$ and an additional 10 measurements are used for testing. Notice that the approach in [7] cannot deal with this situation efficiently since the inputs do not share samples often.

The experiments were done under both the single center and the multi center setting. The plots of task distribution on the left of Fig. 2 suggest that one can get more accurate estimation by using multiple centers. For the multiple center

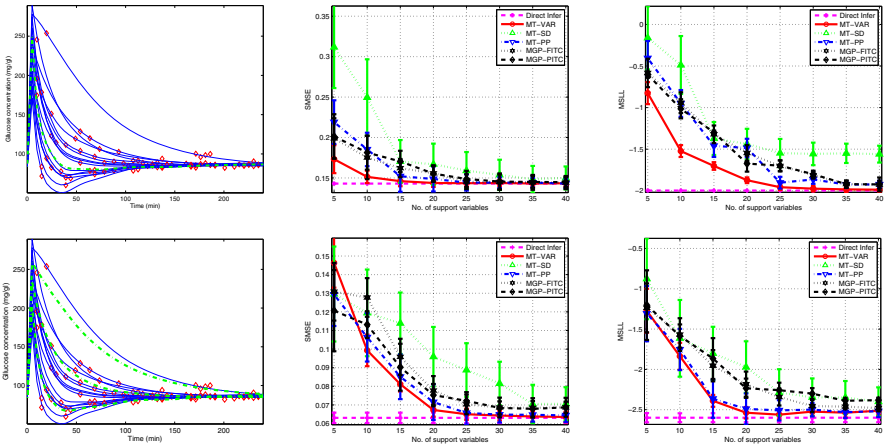


Fig. 2. Simulated Glucose Data. Left: 15 tasks (Blue) with observations (Red Diamonds) and estimated fixed-effect curve (Green) obtained from 1000 IVGTT responses. Center: The average SMSE for all tasks; Right: The average MSLL for all tasks.

case, the number of centers for the proposed method is (arbitrarily) set to be 3 ($K = 3$) and the number of latent function of MGP is set to be 2 ($Q = 2$) (We were not able of obtain reasonable results using MGP when $Q = 3$). The experimental results (top/bottom for single/multi center) are shown in Fig. 2. First, we observe that the multi-center version performs better than the single center one, indicating that the group-based generalization of the traditional mixed-effect model is beneficial. Second, we can see that all the methods achieve reasonably good performance, but that the proposed method significantly outperforms the other methods.

6.3 Real Astrophysics Data

We evaluate our method using the astronomy dataset of [17], where a generative model was developed to capture and classify different types of stars. The dataset, extracted from the OGLEII survey [14], includes stars of 3 types (RRL, CEPH, EB) which constitute 3 datasets in our context. One example of EB is shown in Fig. 3. This star is densely sampled but some stars have less samples and we simulate the sparse case by sub-sampling in our experiments. In [17], we developed a grouped mixed-effect multi-task model that in addition allowed for phase shift of the light measurements. As shown in [17], stars of the same type have a spread of different shapes and the group structure is useful in modeling this domain. However, for inference, [17] used a simple approach clipping sample points to a fine grid of 200 equally spaced points, due to the high dimensionality of the full sample (over 18000 points).

Here we use a random subset of 700 stars (tasks) for each type and preprocess the data normalizing each star to have mean 0 and SD 1, and using universal phasing [8] to phase each time series to align the maximum of a sliding window of 5% of the original points. For each time series, we randomly sample 10 examples for training and 10 examples for testing per evaluation of SMSE and MSLL. The number of centers is set to be 3 for the proposed approach and for MGP we set $Q = 1$ (We were not able to use $Q > 1$). The results for EBs are shown in Fig. (3). We can see that the proposed model outperforms all other methods. For Cepheid and RRL (results not shown due to space limit), the performance

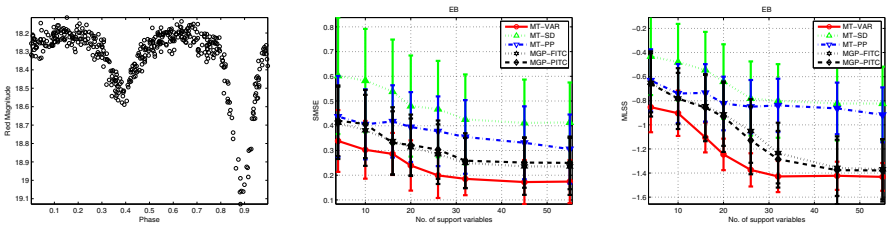


Fig. 3. OGLEII: Left: time series for EB star. Middle and Right show SMSE and MSLL respectively for EB type.

of the proposed model and MGP is very close and they outperform the other methods.

7 Conclusion

The paper develops an efficient variational learning algorithm for the grouped mixed-effect GP for multi-task learning, which compresses the information of all tasks into an optimal set of support variables for each mean effect. Experimental evaluation demonstrates the effectiveness of the proposed method. In future, it will be interesting to derive an online sparse learning algorithm for this model.

Acknowledgement. We would like to thank the authors of [1] who kindly made their code available online. This research was partly supported by NSF grant IIS-0803409. The experiments in this paper were performed on the the Tufts Linux Research Cluster supported by Tufts UIT Research Computing.

References

1. Álvarez, M.A., Lawrence, N.D.: Computationally efficient convolved multiple output Gaussian processes. JMLR 12, 1425–1466 (2011)
2. Álvarez, M.A., Luengo, D., Titsias, M.K., Lawrence, N.D.: Efficient multioutput Gaussian processes through variational inducing kernels. In: AISTATS (2010)
3. Álvarez, M., Rosasco, L., Lawrence, N.: Kernels for vector-valued functions: a review. Arxiv preprint arXiv:1106.6251 (2011)
4. Bonilla, E., Chai, K.M., Williams, C.: Multi-task Gaussian process prediction. In: NIPS, vol. 20, pp. 153–160 (2008)
5. Lu, Z., Leen, T., Huang, Y., Erdogmus, D.: A reproducing kernel Hilbert space framework for pairwise time series distances. In: ICML, pp. 624–631 (2008)
6. Pillonetto, G., De Nicolao, G., Chierici, M., Cobelli, C.: Fast algorithms for nonparametric population modeling of large data sets. Automatica 45(1), 173–179 (2009)
7. Pillonetto, G., Dinuzzo, F., De Nicolao, G.: Bayesian Online Multitask Learning of Gaussian Processes. IEEE T-PAMI 32(2), 193–205 (2010)
8. Protopapas, P., Giammarco, J.M., Faccioli, L., Struble, M.F., Dave, R., Alcock, C.: Finding outlier light curves in catalogues of periodic variable stars. Monthly Notices of the Royal Astronomical Society 369, 677–696 (2006)
9. Quiñonero-Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate gaussian process regression. The Journal of Machine Learning Research 6, 1939–1959 (2005)
10. Rasmussen, C.E., Nickisch, H.: Gaussian Processes for Machine Learning (GPML) Toolbox. JMLR 11, 3011–3015 (2010)
11. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press (2005)
12. Seeger, C., Williams, M., Lawrence, N.: Fast forward selection to speed up sparse gaussian process regression. In: AISTATS 9 (2003)
13. Snelson, E., Ghahramani, Z.: Sparse Gaussian processes using pseudo-inputs. In: NIPS, vol. 18, pp. 1257–1264 (2006)

14. Soszynski, I., Udalski, A., Szymanski, M.: The Optical Gravitational Lensing Experiment. Catalog of RR Lyr Stars in the Large Magellanic Cloud 06. Acta Astronomica 53, 93–116 (2003)
15. Titsias, M.K.: Variational learning of inducing variables in sparse gaussian processes. In: AISTATS (2009)
16. Vicini, P., Cobelli, C.: The iterative two-stage population approach to ivgtt minimal modeling: improved precision with reduced sampling. American Journal of Physiology-Endocrinology and Metabolism 280(1), E179 (2001)
17. Wang, Y., Khardon, R., Protopapas, P.: Shift-invariant grouped multi-task learning for Gaussian processes. In: ECML, pp. 418–434 (2010)
18. Yu, K., Tresp, V., Schwaighofer, A.: Learning Gaussian processes from multiple tasks. In: ICML, pp. 1012–1019 (2005)

Collective Information Extraction with Context-Specific Consistencies

Peter Kluegl[1,2], Martin Toepfer[1], Florian Lemmerich[1], Andreas Hotho[1], and Frank Puppe[1]

[1] Department of Computer Science VI, University of Würzburg, Am Hubland, Würzburg, Germany
[2] Comprehensive Heart Failure Center, University of Würzburg, Straubmühlweg 2a, Würzburg, Germany
{pkluegl,toepfer,lemmerich,hotho,puppe}@informatik.uni-wuerzburg.de

Abstract. Conditional Random Fields (CRFs) have been widely used for information extraction from free texts as well as from semi-structured documents. Interesting entities in semi-structured domains are often consistently structured within a certain context or document. However, their actual compositions vary and are possibly inconsistent among different contexts. We present two collective information extraction approaches based on CRFs for exploiting these context-specific consistencies. The first approach extends linear-chain CRFs by additional factors specified by a classifier, which learns such consistencies during inference. In a second extended approach, we propose a variant of skip-chain CRFs, which enables the model to transfer long-range evidence about the consistency of the entities. The practical relevance of the presented work for real-world information extraction systems is highlighted in an empirical study. Both approaches achieve a considerable error reduction.

Keywords: information extraction, conditional random fields, collective, context-specific consistencies, long-range dependencies.

1 Introduction

The accurate transformation of unstructured data into a structured representation for further processing is an active area of research with many interesting challenges. One central task for mining unstructured textual data is Information Extraction (IE), which tries to find well-defined entities and relations in textual data. Over the last decade, statistical sequence labeling models and especially Conditional Random Fields (CRFs) [10] became the dominant technique for IE tasks. CRFs are discriminative undirected probabilistic graphical models often trained in a supervised fashion. When applied on textual data, they are usually designed as a linear chain with the first order Markov assumption.

In many scenarios, the entities in textual data are not independent and identically distributed. Recently, much effort went in new approaches that can be

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 728–743, 2012.

summarized under the term Collective IE [2,4,8,9,15]. They break the linear-chain assumption and model also long-range dependencies in order to label related entities or instances collectively. One example is Named Entity Recognition (NER), a task that aims at the extraction of persons or similar entities. Here, the accuracy can be improved by the assumption that similar tokens should have the same label or by providing contextual evidence of related tokens.

In semi-structured documents a different form of long-range dependency often occurs. Here, the context in which the textual data is created or written introduces a homogeneous composition of the entities. The reference section of this paper, for example, is generated using a style guide that defines the layout of the citation information. Thus, all author entities end with a colon. However, the reference sections of other publications follow different style guides in which the author possibly ends with a period. Another example for consistency introduced in a certain context is curricula vitae: Each author describes his or her employments homogeneously but possibly with an arrangement of the interesting entities different from other authors. If these long-range dependencies are not taken into account, then the IE system faces a heterogeneous and inconsistent composition of the entities in the complete dataset. However, by considering the similarities of entities within a context and processing those entities collectively, many labeling errors can be prevented. The accuracy for the detection of the author of a reference, for example, can be greatly increased when the model is encouraged that all authors in a reference section should end identically.

In this work, we present two collective IE approaches based on CRFs that are able to exploit such context-specific consistencies. Both approaches consult a classifier, which detects consistent boundaries of an entity within one context. This classifier is trained during inference on an intermediate label sequence predicted by an additional model. The generalization of the classifier's learning algorithm detects only equally shaped boundaries ignoring entities that break the consistency assumption. This evidence about the consistency is exploited in two different models. The first model extends linear-chain CRFs with additional unigram factors. The positions of the factors are given by the classification result of the classifier combined with the predicted label sequence. In a second approach, we investigate a variant of skip-chain CRFs [15]. Instead of adding dependencies for similar tokens, the boundaries of related entities are connected. These additional edges then transport evidence about the consistency of the entities' compositions at the positions indicated by the classifier. In an empirical study, we evaluate our approaches with real-word datasets, for the segmentation of references and for template extraction in curricula vitae. The results show the practical relevance of the presented work for real-world IE systems. Our approaches are able to achieve a substantial error reduction, up to 34%.

The rest of the paper is structured as follows: In Section 2, we recap different variants of CRFs for information extraction. The two novel approaches for exploiting context-specific consistencies are described in Section 3. Their results in an empirical study are presented in Section 4. Section 5 gives a short overview of the related work and Section 6 concludes with a summary.

2 Conditional Random Fields

Conditional Random Fields (CRFs) [10] are undirected graphical models which model conditional distributions over random variables $\mathbf{y}$ and $\mathbf{x}$. Given exponential potential functions $\Phi(\mathbf{y_c}, \mathbf{x_c}) = \exp\left(\sum_k \lambda_k f_k(\mathbf{y_c}, \mathbf{x_c})\right)$ a CRF assigns

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \Phi(\mathbf{y_c}, \mathbf{x_c}) \tag{1}$$

to a graph with cliques $\mathcal{C}$ under model parameters $\theta = (\lambda_1, \ldots, \lambda_K) \in \mathbb{R}^K$. The partition function $Z(\mathbf{x}) = \sum_{y'} \prod_{c \in \mathcal{C}} \Phi(\mathbf{y_c}, \mathbf{x_c})$ is a normalization factor to assert $\sum_y p_\theta(\mathbf{y}|\mathbf{x}) = 1$. The feature functions f_k can be real valued in general, however, we assume binary feature functions if not mentioned differently.

When CRFs are applied for IE tasks, the model is adapted to the properties of sequential data or textual documents respectively. Therefore the graph structure is normally restricted to be a linear chain representing the sequence of labels that are assigned to a sequence of tokens. The entities of the IE tasks are identified by sequences of equal labels. If linear-chain CRFs also model long-range dependencies with additional edges between distant labels, then the models are called skip-chain CRFs [15]. Both models are shortly outlined in the following.

2.1 Linear-Chain CRFs

Linear chain CRFs [10] restrict the underlying graph structures to be linear sequences, typically with a first order Markov assumption. The assignment of y_t given $\mathbf{x}$ and $\mathbf{y} - y_t = (y_t)_{t=1,\ldots,t-1,t+1,\ldots,T}$ is then only dependent on y_{t-1},y_t, y_{t+1} and $\mathbf{x}$. The probability of a label sequence $\mathbf{y}$ given an token sequence $\mathbf{x}$ is modeled by

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Phi_{\text{L}}(y_t, y_{t-1}, \mathbf{x}). \tag{2}$$

We are using Φ_{L} to describe the factors of the linear-chain edges that link adjacent labels:

$$\Phi_{\text{L}}(y_t, y_{t-1}, \mathbf{x}) = \exp\left\{\sum_k \lambda_{\text{L}k} f_{\text{L}k}(y_t, y_{t-1}, \mathbf{x}, t)\right\}. \tag{3}$$

The discriminative impact of the feature functions $f_{\text{L}k}$ is weighted by the parameters $\theta = \theta_{\text{L}} = \{\lambda_{\text{L}k}\}_{k=1}^{K}$. The feature functions can typically be further factorized into indicator functions $p_{\text{L}k}$ and observation functions $q_{\text{L}k}$

$$f_{\text{L}k}(y_t, y_{t-1}, \mathbf{x}, t) = p_{\text{L}k}(y_t, y_{t-1}) \cdot q_{\text{L}k}(\mathbf{x}, t). \tag{4}$$

$p_{\text{L}k}$ returns 1 for a certain label configuration and $q_{\text{L}k}$ relies only on the input sequence $\mathbf{x}$. Thus, a feature function, e.g., that indicates capitalized tokens, can be separately weighted for each label transition. Figure 1 contains an example

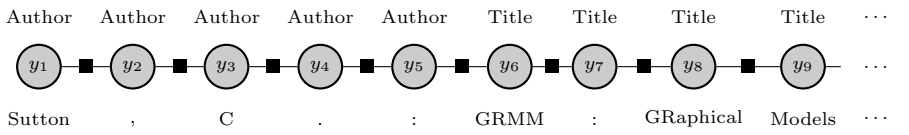


Fig. 1. A linear-chain CRF applied on the reference segmentation task, i.e., the 14th reference of this paper. The associated labels and tokens are depicted above and below the variables.

of a linear-chain CRF in factor graph representation, which is applied for the reference segmentation task. We added the label and token sequence for better understanding. Dependencies of the factors to tokens are omitted for simplicity.

2.2 Skip-Chain CRFs

Skip-chain CRFs [15] break the first order Markov assumption of linear-chain CRFs by adding potentials to the graph that address dependencies between distant labels and tokens. A set $I_{\mathrm{X}} = \{(u, v)\} \subset \{1, \ldots, T\} \times \{1, \ldots, T\}$ defines positions u, v for which y_u, y_v are connected by skip edges. We refer to components of skip-chain CRFs with the index X in order to point out their usage in previous publications, e.g., [15]. The set I_{X} unrolls skip edges based on token similarity and is therefore only dependent on the token sequence $\mathbf{x}$. In NER tasks, for example, the accuracy can often be increased when the model is encouraged to label similar tokens identically. For controlling the computational cost, I_{X} has to be kept small. An extension of Equation 2 with additional skip edges results in the conditional probability

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Phi_{\mathrm{L}}(y_t, y_{t-1}, \mathbf{x}) \prod_{(u,v)\in I_{\mathrm{X}}} \Psi_{\mathrm{X}}(y_u, y_v, \mathbf{x}). \tag{5}$$

The potentials Ψ_{X} for the skip edges are given by

$$\Psi_{\mathrm{X}}(y_u, y_v, \mathbf{x}) = \exp\left\{\sum_k \lambda_{\mathrm{X}k} f_{\mathrm{X}k}(y_u, y_v, \mathbf{x}, u, v)\right\} \tag{6}$$

extending the complete set of parameters $\theta = \theta_{\mathrm{L}} \cup \theta_{\mathrm{X}}$. The feature functions factorize again in an indicator function $p_{\mathrm{X}k}$ and an observation function $q_{\mathrm{X}k}$:

$$f_{\mathrm{X}k}(y_u, y_v, \mathbf{x}, u, v) = p_{\mathrm{X}k}(y_u, y_v, u, v) \cdot q_{\mathrm{X}k}(\mathbf{x}, u, v) \tag{7}$$

The observation function enables the model to share observed information between the positions u and v and their neighborhoods, e.g., for providing local evidence at a position where such information is missing.

3 CRFs with Context-Specific Consistencies

This section introduces two different approaches to exploit context consistencies. Both methods are divided into two different parts. When unrolling the graph during inference, we first have to detect the patterns that describe the consistency of the context. Secondly, we need to incorporate the gained knowledge into the graph structure for a better prediction. In the following, we first describe challenges of dependencies on the label sequence and the applied method to learn context-specific consistencies. Then, we explain the differences of the two approaches which only concern the structure and complexity of the models that exploit the context-specific patterns.

3.1 Context-Specific Consistencies

Context-specific consistencies refer to a special kind of long-range dependencies that are often found in semi-structured documents. The interesting entities within a specific context or document share a similar composition caused by the process the document is created or written in. Examples for this process are authors that arrange the entities homogeneously or templates that enforce a specific layout. We call these consistencies context-specific, because the actual composition is unknown at application time and can strongly vary between contexts. There are many different ways to describe the composition of entities. In this work we take a closer look at the entity boundaries, that is, the first and the last label of the entity[1]. Other possibilities include the labels within the boundaries of an entity. More generally, this can be extended to any kind of label transition. However, the boundaries alone are very suitable to classify an entity independently of the actual label transition and allow to restrict the long-range dependencies to a minimal amount.

3.2 Dependencies Based on the Label Sequence

In this paper, we investigate how these consistencies can be exploited with the idea of skip-chain CRFs or in general CRFs with additional potentials for long-range dependencies. In contrast to skip-chain CRFs, where the potentials are only based on the token sequence (cf. Equation 6), our additional potentials are mainly dependent on the label sequence. Our approaches need a prediction of the assignment in order to be able to link or relate the boundaries of the entities. The label sequence (hidden variables) is of course not available during inference when we unroll the graph on an instance with all potentials since it is the result of the computation of $p_\theta(\mathbf{y}|\mathbf{x})$. However, there are many different ways to provide a prediction of the label sequence during inference. Our initial choice was to incrementally unroll the graph: We first unrolled the potentials of the linear-chain part, computed the currently most likely label sequence and used

[1] No additional encoding like IOB is applied in order to identify the entities in the label sequence.

this prediction to further unroll the additional potentials. However, we observed problems with the parameter estimation and inference mechanism applied in this work (cf. Section 3.6). While we sometimes achieved remarkable improvements, the approach frequently did not converge at all. Therefore, we utilize a separate static linear-chain model in order to provide a constant prediction of the label sequences, which corresponds to the approach of stacked graphical models [8,9,7]. Here, an initial model is used to compute new features for a stacked model. In our approach, however, the predicted assignments of the initial model lead to additional potentials. Normally, cross-fold training is applied for the initial model in order to prevent unrealistic predictions during training of the stacked model. We neglect this improvement in the belief that the advantages of the presented models prevail.

3.3 Learning Context-Specific Consistencies

When we try to exploit the context-specific consistencies, it is very helpful to acquire a description or model for the consistencies in each context or document. Thereby, one can distinguish consistent and inconsistent boundaries of the entities. As in previous work [7], we train and apply a binary classifier on the boundaries of an entity within one context. The learning task of the classifier for a boundary of one type of entity is defined as following: Each token of the context is a training example and the features of the CRF become binary attributes, possibly with an additional windowing. The intermediate label sequence (cf. 3.2), respectively the predicted boundaries, specifies the learning target of the classifier. The generalization capacity of the classifier's learning algorithm is the key to gain knowledge about the context-specific consistency. We assume that the hypothesis space of the classifier is not sufficient to provide a perfectly accurate model and therefore only describes the dominant consistency.

A suitable classifier for the tasks presented in this work has to provide following properties:

- The classifier should be efficient with respect to its execution time since it is trained and applied on all emerging label sequences during inference.
- The classifier should not tend to overfit since it is trained and applied on possibly erroneous data. These errors should not be reproduced. In general, overfitting can also be restrained by limiting the amount of attributes.
- The classifier should not combine different hypotheses in order to solve the classification problem if only one consistency for the boundary exists in data as it is in our examples.
- The classifier should handle label bias correctly, even if there are only a few true positives and thousands of true negatives.

We decided to utilize a simple but efficient rule learner based on subgroup discovery [6], an exhaustive search for the best conjunctive pattern describing an target attribute, respectively the entity's boundary. This technique fulfills all requirements with minimal efforts of configuration and is fast enough if the set

of attributes is constrained. As an improvement to [7], a new quality function F_1^{exp} selects the best pattern:

$$F_1^{exp} = \frac{2 \cdot tp}{2 \cdot tp + fn + fp} \cdot \left(1 - \left(\frac{|tp + fp - E_y|}{\max(tp + fp, E_y)}\right)^2\right) \tag{8}$$

The left part of this measure describes the traditional F_1-Measure, that is how well the pattern reproduces the predicted boundaries. The right factor is a penalty term for the divergence of the amount of instances classified as boundaries to a given variable E_y, the expected amount of boundaries in a context. E_y can simply be estimated using the token sequence and the feature functions in the data set applied in this work. In the domain of reference segmentation, for example, we expect that each reference contains exactly one author. Although this is not true in general, it provides for a valuable weighting of the hypothesis space and further reduces overfitting.

3.4 Comb-Chain CRFs

In a first approach, we extend the variables of a linear chain model with additional (unigram) factors dependent on the classification result (cf. Figure 2). Hence, we chose the name comb-chain CRFs for this approach because of the layout of the graph.

Let $\mathcal{R}_b(y)$ and $\mathcal{R}_e(y)$ be the set of positions, which are identified by the classifier as the beginning and end of an entity with the label y. We can now define the positions of additional factors:

$$\begin{aligned} \mathcal{U}_b &= \left\{ u : y_{u-1} \neq y_u \vee u \in \bigcup_y \mathcal{R}_b(y) \right\} \\ \mathcal{U}_e &= \left\{ u : y_u \neq y_{u+1} \vee u \in \bigcup_y \mathcal{R}_e(y) \right\} \\ \mathcal{U} &= \mathcal{U}_b \cup \mathcal{U}_e \end{aligned} \tag{9}$$

$\mathcal{U}_b$ and $\mathcal{U}_e$ contain all positions that are either intermediately labeled by the external model or are identified by the classifier as the beginning, respectively end of an entity. The conditional probability is then defined as[2]

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Phi_{\mathrm{L}}(y_t, y_{t-1}, \mathbf{x}) \prod_{u \in \mathcal{U}} \Psi_{\mathrm{C}}(\mathbf{y}, u) \tag{10}$$

and the potentials for the unigram factor are given by

$$\Psi_{\mathrm{C}}(\mathbf{y}, u) = \exp\left\{ \sum_k \lambda_{\mathrm{C}k} f_{\mathrm{C}k}(\mathbf{y}, u) \right\} \tag{11}$$

[2] The different usage of $\mathbf{y}$ for the predicted sequence and the label configuration of the parameters deduces from the context.

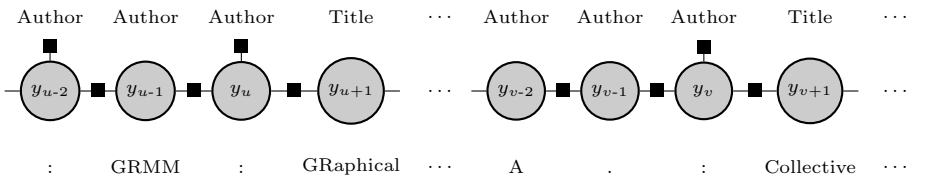


Fig. 2. An excerpt of a comb-chain graph with erroneous labeling whereas only additional factors for the end of the author are displayed. The output functions indicate a missing end at position $y_{u\text{-}2}$, a surplus end at y_u and a consistent end at y_v.

whereas $\theta_{\text{C}} = \{\lambda_{\text{C}k}\}$ is the set of additional parameters for the classifier template. We let the feature function factorize into an indicator function $p_{\text{C}k}$ and an output function $q_{\text{C}k}$:

$$f_{\text{C}k}(\mathbf{y}, u) = p_{\text{C}k}(y_u) \cdot q_{\text{C}k}(\mathbf{y}, u) \tag{12}$$

We introduce six different output functions:

$$\begin{aligned}
q_{\text{e-consistent}}(\mathbf{y}, u) &= \begin{cases} 1 & \text{iff } y_u \neq y_{u+1} \wedge u \in \mathcal{R}_e(y_u) \\ 0 & \text{else} \end{cases} \\
q_{\text{e-project}}(\mathbf{y}, u) &= \begin{cases} 1 & \text{iff } y_u \neq \tilde{y} \wedge u \in \mathcal{R}_e(\tilde{y}) \\ 0 & \text{else} \end{cases} \\
q_{\text{e-suppress}}(\mathbf{y}, u) &= \begin{cases} 1 & \text{iff } y_u \neq y_{u+1} \wedge u \notin \mathcal{R}_e(y_u) \\ 0 & \text{else} \end{cases}
\end{aligned} \tag{13}$$

The output functions $q_{\text{b-consistent}}$, $q_{\text{b-project}}$ and $q_{\text{b-suppress}}$ are defined equivalently for the beginning of an entity. This reflects the meaning, that is the result of the classification combined with the intermediate labeling: $q_{\text{e-consistent}}$ indicates a true positive, $q_{\text{e-project}}$ a false positive and $q_{\text{e-suppress}}$ a false negative classification compared to the label sequence. Together these feature functions supply evidence, which parts of the label sequence agree with the consistency and which parts should be altered in order to gain a higher likelihood. The resulting graph of the model contains no loops and provides therefore less challenges for an inference mechanism.

The idea of comb-chain CRFs is summarized with an example for the segmentation of references (cf. Figure 2). Let the reference section of this paper be the input sequence. When unrolling the graph, we ask the external model for an intermediate labeling specifying the entities. A classifier is trained to detect the boundaries of the entities. The descriptive result of the classifier for the end of the author is, for example, a pattern like "A period followed by a colon". Now, the additional potentials with the output functions influence the model to assign a high likelihood to label sequences that confirm with the description of the classifier.

3.5 Skyp-Chain CRFs

Skyp-chain CRFs are a variant of skip-chain CRFs. But instead of creating additional edges between labels whose tokens are similar or identical, this approach adds long-range dependencies based on the patterns occurring in the predicted label sequence $\mathbf{y}$ and the classification result. Thus, the small modification of the name. When applying skyp-chain CRFs for exploiting context-specific consistencies, two additional differences to published approaches for skip-chain CRFs or similar collective IE models can be identified:

1. There is no need to transfer local evidence to distant labels since we already assume a homogeneous composition of the entities.
2. Useful observation functions for the skip edges cannot be specified, because the relevance of certain properties is unknown.

We first define the set of additional edges that specify the positions of the long-range dependencies using the positions $\mathcal{U}_b$ and $\mathcal{U}_e$ of Equation 9.

$$\begin{aligned} \mathcal{E}_b &= \{(u,v) : u \neq v \wedge y_u = y_v \wedge u \in \mathcal{U}_b \wedge v \in \mathcal{U}_b\} \\ \mathcal{E}_e &= \{(u,v) : u \neq v \wedge y_u = y_v \wedge u \in \mathcal{U}_e \wedge v \in \mathcal{U}_e\} \\ \mathcal{E} &= \mathcal{E}_b \cup \mathcal{E}_e \end{aligned} \tag{14}$$

The set $\mathcal{E}_b$ contains edges that connect the start label of an entity with all other start labels of entities with the same type. The set $\mathcal{E}_e$ refers accordingly to the links between the end labels of entities. Further, we introduce a parameter m_e for controlling the model complexity that restricts the maximal amount of additional long-range dependencies for each variable. E.g., for $m_e = 2$, a label is only connected to the closest previous and following boundary of the same entity type.

Our skyp-chain approach extends the linear-chain model with additional potentials for edges defined in Equation 14. The conditional probability for the assignment of the label sequence is given by

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Phi_{\mathrm{L}}(y_t, y_{t-1}, \mathbf{x}) \prod_{(u,v)\in\mathcal{E}} \Psi_{\mathrm{Y}}(\mathbf{y}, u, v)\,. \tag{15}$$

An example of an unrolled graph of this model is depicted in Figure 3. Similar to Equation 6, the additional potentials factorize to

$$\Psi_{\mathrm{Y}}(\mathbf{y}, u, v) = \exp\left\{\sum_k \lambda_{\mathrm{Y}k} f_{\mathrm{Y}k}(\mathbf{y}, u, v)\right\}, \tag{16}$$

resulting in the complete parameter set $\theta = \theta_{\mathrm{L}} \cup \theta_{\mathrm{Y}}$ with $\theta = \theta_{\mathrm{Y}} = \{\lambda_{\mathrm{Y}k}\}$ to be estimated for this model. In contrast to the skip-chain model, our feature functions depend on the complete (predicted) label sequence $\mathbf{y}$. The feature functions consist again of an indicator function for the label configuration, but

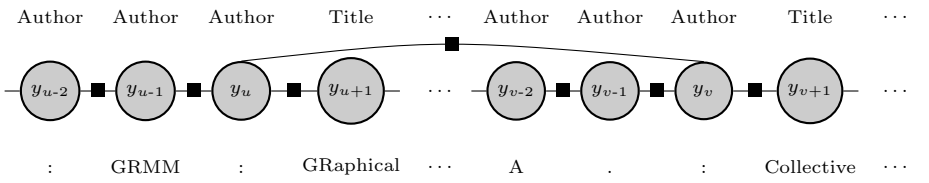


Fig. 3. An excerpt of a skyp-chain graph with erroneous labeling. Only one additional edge for the end of the author is displayed. The likelihood of the sequence is decreased because only position $u-2$ and v but not u were identified as a boundary by the classifier.

not of an observation function on the input sequence. Instead we apply the output functions of Equation 13 separately for the source and destination of the skip edge.

Let us illustrate the skyp-chain model in an example for reference segmentation (cf. Figure 3). Let the input sequence be the reference section of this paper. When the graph of the model is unrolled during inference, the most probable label assignments are calculated. During this process we consider long-range dependencies, e.g., for the end of the author entities (cf. labels y_u and y_v in Figure 3). Due to our additional potentials, label sequences with boundaries that are identified by the classifier as consistently structured become more likely. In Figure 3, the likelihood of the sequence is decreased in comparison to a graph with an additional edge between the labels y_{u-2} and y_v.

3.6 Parameter Estimation and Inference

We compute $p_\theta(\mathbf{y}|\mathbf{x})$ to decide which label sequence $\mathbf{y}$ is most likely for the observed token sequence $\mathbf{x}$, and to estimate the parameters θ of the model. The applied inference technique, tree based reparameterization (TRP) [17], is related to belief propagation and computes approximate marginals for loopy graphs. TRP is also used in [15] for the original skip-chain models. Unfortunately, severe convergence problems could be oberserved when applied on complex graph structures. The parameters θ of our models are obtained using training data $D = \left\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right\}_{i=1}^{N}$ and maximum a-rrposteriori estimation. The log likelihood $\mathcal{L}(\theta|D)$ of the model parameters given the training examples is optimized with the quasi-Newton method L-BFGS and a Gaussian prior on the parameters as in [15].

4 Experimental Results

We demonstrate the advantages of the presented approach in a five-fold cross evaluation in two different real-world applications: The segmentation of references and the template extraction in curricula vitae. First, both domains and the real-world datasets are described and then we specify the settings of the evaluation. Finally, we present and discuss the empirical results.

4.1 Datasets

Two datasets are utilized in the evaluation of this work. The dataset *References* origins in a domain that is very popular for the evaluation of novel IE techniques (cf. [1,11,12,13]), whereas the dataset *Curricula Vitae* belongs to classical IE problems of template extraction.

References. This dataset for the segmentation of references was introduced in previous work [7] and consists only of complete reference sections of real publications, mainly from the computer science domain. The application behind this dataset consists mainly in the identification of Bibtex fields in crawled publications, which can be used to improve scientific search engines or to analyze citation graphs. The dataset contains 566 references in 23 reference sections with overall 15 different labels and is comparable to datasets of previous publications with respect of size, label and feature set, e.g., Peng et al. [11]. For the evaluation in this paper, we reduced the label set for the identification of the entities AUTHOR, DATE, TITLE and VENUE, which are sufficient for the targeted application. The dataset can be freely downloaded[3]. We skip a detailed description of the features and refer to the archive because it contains all applied features.

Curricula Vitae. The IE task of this dataset is to identify the time span and company for which the author of these documents worked in a stage of his or her life (employments). This information can be used to improve the search for suitable future employees for certain projects. The data set consists of 68 curricula vitae and is annotated with 896 companies or sectors[4] and 937 time spans in overall 921 stages of life. We use the label DATE for the time span and the label CLIENT for the companies or sectors. The feature set extends the feature set of the dataset *References* with additional domain-specific features like the number of the line, the position within a line and keywords for company prefixes/suffixes and date indicators. Unfortunately, we can not publish this dataset due to non-disclosure agreements.

4.2 Evaluation Measure

The performance of the presented models is measured with the F_1 score. Let tp be the number of true positive labeled tokens and fn and fp respectively the number of false negatives and false positives tokens. *Precision*, *recall* and F_1 are then defined as:

$$precision = \frac{tp}{tp + fp}, \quad recall = \frac{tp}{tp + fn}, \quad F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}.$$

For the dataset *References* we present the F_1 score combined for all labels, whereas we distinguish the labels DATE and CLIENT for *Curricula Vitae*.

[3] http://www.is.informatik.uni-wuerzburg.de/staff/kluegl_peter/research/

[4] The authors of the curricula vitae sometimes anonymize the actual name of a company and replace it with the sector in which the company is located.

Table 1. F_1 scores for the segmentation of references

References	
	ALL
LINEAR CHAIN	0.966
COMB CHAIN	0.976
SKYP CHAIN	0.972

Table 2. F_1 scores for template extraction in curricula vitae

Curricula Vitae		
	DATE	CLIENT
LINEAR CHAIN	0.944	0.725
COMB CHAIN	0.962	0.814
SKYP CHAIN	0.962	0.764

4.3 Settings

All models are trained with identical settings and features. In order to minimize the model complexity of the skyp-chain approach, we set $m_e = 2$. We used 11 (for *References*) and 12 (for *Curricula Vitae*) manually selected feature functions in a window of five tokens as attributes for the rule learner. The learned descriptions had a maximum of three attributes and a minimum quality score of 0.01. For the dataset *References*, only the boundaries for the labels AUTHOR, DATE, TITLE are considered. Our implementation of the CRFs is based on the GRMM package [14].

4.4 Results

We compare the proposed models to a linear-chain CRF (base line). Additionally, we have applied the stacked approach with exact inference of [7] for a comparable model. However, its evaluated F_1 score was surprisingly lower than our base line. An analysis revealed that the different implementations of CRFs and the varying definition of an instance influenced the results. We have also considered different variants of skip-chain CRFs, but none of them returned noteworthy results. As a consequence, we compare our models only with the base line.

The results of the five-fold cross evaluation are depicted in Table 1 for the dataset *References* and in Table 2 for the dataset *Curricula Vitae*. The comb-chain models achieve overall an average error reduction of over 30% and increased the measured averaged F_1 score by at least 1%, 9% for the label CLIENT. The skyp-chain model provides more challenges for the inference technique and is only able surpass the comb-chain results for the label DATE of the dataset *Curricula Vitae*. In the evaluation of the remaining label, the average error reduction is 14%.

If the comb-chain model is compared to the skyp-chain model, then it becomes apparent that the skyp-chain model with the applied inferencing technique TRP has no advantages when exploiting consistencies even at the cost of a computationally more expensive inference. Table 3 and Table 4 contain the average evaluation time for one fold. In general, it takes longer to train models with the larger dataset *Curricula Vitae*.

Table 3. Average time for one fold (*References*)

References	
Linear Chain	0.03h
Comb Chain	0.17h
Skyp Chain	0.53h

Table 4. Average time for one fold (*Curricula Vitae*)

Curricula Vitae	
Linear Chain	0.11h
Comb Chain	0.27h
Skyp Chain	0.97h

4.5 Discussion

The evaluated results of the presented IE models have a valuable influence on real-world applications. An error reduction of 30% considerably improves the quality of automatically extracted entities in the database and reduces the workload to correct possible IE errors. The reported increase of the accuracy and the corresponding error reduction of the presented models compete well with published approaches for Collective IE, joint inference in IE or other models that exploit long-range dependencies.

The performance time of the presented models is in our opinion fast enough for the planned applications, but can still be increased with further optimizations or faster inference and learning techniques.

5 Related Work

In this section, we give a short overview of the related work, which can be categorized into Information Extraction (IE) publications about:

- Collective IE for Named Entity Recognition (NER).
- Collective IE with respect to structured texts.
- Collective IE with context-specific consistencies.
- Improved IE models in general, evaluated for the segmentation of references.

Collective IE is an active and popular field of research and thus we can only discuss some representatives of each category.

Models of collective approaches for NER are often motivated by two assumptions: The labeling of similar tokens is quite consistent within a given context or document since those mentions mostly refer to the same type of entity. The discriminative features to detect the entities are sparsely distributed over the document. Thus, the accuracy for different mentions of an entity can be improved by leveraging and transferring their local context to distant positions.

Bunescu et al.[2] use Relational Markov Networks and model dependencies between distant entities. They apply special templates in order to assign equal labels if the text of the tokens is identical. The skip-chain approach introduced by Sutton et al.[15] extends linear-chain CRFs with additional factors for long-range dependencies. They link the labels of similar tokens and provide feature functions that combine evidence of both positions by which missing context can

be transferred. Finkel et al. [4] criticize the usage of believe propagation and apply Gibbs sampling for enforcing label consistency and extraction template consistency constraints. All of these approaches with higher order structures fight the exponential increase in model complexity and are forced to apply approximate inference techniques instead of exact algorithms. Kou et al. [8] and Krishnan et al. [9] have shown that stacked graphical models with exact inference can compete with the accuracy of those complex models. They reduce the computational cost by applying an ensemble for two linear-chain CRFs where they aggregate the output of the first models in order to provide information about related instances or entities to a stacked model.

Yang et al. [19] and Gulhane et al. [5] presented work about IE in webforums and websites. The first approach applies Markov Logic Networks to encode properties of a typical forum page like attribute similarities among different posts and sites. The second approach developed an Apriori-style algorithm and assumes that values of an attribute distributed over different pages are similar for equal entities and the pages of one website share a similar structure due to the creation template. In contrast to our models, both approaches are domain-dependent and rely on prior knowledge about the structure.

In previous work [7], we proposed stacked CRFs in combination with rule learning techniques to exploit context-specific consistencies. The output of the first CRF was utilized to learn the manifestation of feature functions for the stacked CRF. The approach was evaluated only for the segmentation of references and achieved a significant error reduction compared to a linear-chain CRF. The stacked CRFs with feature induction during inference is similar to the comb-chain model. However, we developed a novel quality function and utilize the classification result to add new potentials instead of only normal features for a label transition. The skyp-chain approach further increases the model complexity and adds edges for long range dependencies.

The segmentation of references is a widely used domain for the evaluation of novel machine learning and IE models. The work of Peng et al. [11] provides a deep analysis of different settings and established linear-chain CRFs as the state-of-the-art for the segmentation of references. Approaches for joint inference [12,13] combine different tasks within a model. Here, the accuracy of the labeling can be increased when entity resolution and segmentation are jointly performed. Finally, Bellare et al. [1] present a semi-supervised approach for reference segmentation by encoding expectations in higher-order constraints that cover more expressive and structural dependencies than the underlying model.

6 Conclusions

Exploiting context-specific consistencies can substantially increase the accuracy of sequence labeling in semi-structured documents. We presented two approaches based on CRFs, which combine ideas of stacked graphical models and higher-order models like skip-chain CRFs. Both approaches outperform the common models and have a valuable impact for real-world IE applications. The

comb-chain CRFs are able to achieve an average error reduction of about 30% in two datasets.

For future work, two interesting improvements can be identified: On a technical level, the usage of newer inference techniques for factor graphs like SampleRank [18] should be able to avoid some of the described problems. On a more conceptual level, a joint inference approach like [13] that combines labeling and consistency identification within a probabilistic graphical model has the potential to gain further advantages in the evaluated domains.

Acknowledgments. This work was supported by the Competence Network Heart Failure, funded by the German Federal Ministry of Education and Research (BMBF01 EO1004).

References

1. Bellare, K., Druck, G., McCallum, A.: Alternating Projections for Learning with Expectation Constraints. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in AI, pp. 43–50. AUAI Press (2009)
2. Bunescu, R., Mooney, R.J.: Collective Information Extraction with Relational Markov Networks. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, ACL 2004. Association for Computational Linguistics, Stroudsburg (2004)
3. Elidan, G., McGraw, I., Koller, D.: Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing. In: Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI), Boston, Massachussetts (July 2006)
4. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local Information into Information Extraction Systems by Gibbs Sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL 2005, pp. 363–370. Association for Computational Linguistics, Stroudsburg (2005)
5. Gulhane, P., Rastogi, R., Sengamedu, S.H., Tengli, A.: Exploiting Content Redundancy for Web Information Extraction. Proc. VLDB Endow. 3, 578–587 (2010)
6. 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)
7. Kluegl, P., Toepfer, M., Lemmerich, F., Hotho, A., Puppe, F.: Stacked Conditional Random Fields Exploiting Structural Consistencies. In: Carmona, P.L., Sánchez, J.S., Fred, A. (eds.) Proceedings of 1st International Conference on Pattern Recognition Applications and Methods (ICPRAM), pp. 240–248. SciTePress, Vilamoura (2012)
8. Kou, Z., Cohen, W.W.: Stacked Graphical Models for Efficient Inference in Markov Random Fields. In: Proceedings of the 2007 SIAM Int. Conf. on Data Mining (2007)
9. Krishnan, V., Manning, C.D.: An Effective two-stage Model for Exploiting non-local Dependencies in Named Entity Recognition. In: Proc. of the 21st Int. Conf. on Computational Linguistics and the 44th Annual Meeting of the ACL. ACL-44, pp. 1121–1128. ACL, Stroudsburg (2006)
10. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proc. 18th International Conf. on Machine Learning, pp. 282–289 (2001)

11. Peng, F., McCallum, A.: Accurate Information Extraction from Research Papers using Conditional Random Fields. In: HLT-NAACL, pp. 329–336 (2004)
12. Poon, H., Domingos, P.: Joint Inference in Information Extraction. In: AAAI 2007: Proceedings of the 22nd National Conference on Artificial Intelligence, pp. 913–918. AAAI Press (2007)
13. Singh, S., Schultz, K., McCallum, A.: Bi-directional Joint Inference for Entity Resolution and Segmentation Using Imperatively-Defined Factor Graphs. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 414–429. Springer, Heidelberg (2009)
14. Sutton, C.: GRMM: GRaphical Models in Mallet (2006), http://mallet.cs.umass.edu/grmm/
15. Sutton, C., McCallum, A.: Collective Segmentation and Labeling of Distant Entities in Information Extraction. In: ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields (2004)
16. Sutton, C.A., McCallum, A.: Improved Dynamic Schedules for Belief Propagation. In: Parr, R., van der Gaag, L.C. (eds.) UAI, pp. 376–383. AUAI Press (2007)
17. Wainwright, M.J., Jaakkola, T., Willsky, A.S.: Tree-based Reparameterization for Approximate Inference on Loopy Graphs. In: NIPS, pp. 1001–1008 (2001)
18. Wick, M.L., Rohanimanesh, K., Bellare, K., Culotta, A., McCallum, A.: SampleRank: Training Factor Graphs with Atomic Gradients. In: Getoor, L., Scheffer, T. (eds.) ICML, pp. 777–784. Omnipress (2011)
19. Yang, J.M., Cai, R., Wang, Y., Zhu, J., Zhang, L., Ma, W.Y.: Incorporating Site-level Knowledge to Extract Structured Data from Web Forums. In: Proceedings of the 18th International Conference on World Wide Web, pp. 181–190. ACM (2009)

Supervised Learning of Semantic Relatedness

Ran El-Yaniv and David Yanay

Department of Computer Science, Technion - Israel Institute of Technology
rani@cs.technion.ac.il, dudu.yanay@gmail.com

Abstract. We propose and study a novel supervised approach to learning statistical semantic relatedness models from subjectively annotated training examples. The proposed semantic model consists of parameterized co-occurrence statistics associated with textual units of a large background knowledge corpus. We present an efficient algorithm for learning such semantic models from a training sample of relatedness preferences. Our method is corpus independent and can essentially rely on any sufficiently large (unstructured) collection of coherent texts. Moreover, the approach facilitates the fitting of semantic models for specific users or groups of users. We present the results of extensive range of experiments from small to large scale, indicating that the proposed method is effective and competitive with the state-of-the-art.

1 Introduction

Semantic relatedness (SR) has been steadily gaining attention among statistical NLP and AI researchers. The interest in SR research has been reinforced by the emergence of applications that can greatly benefit from SR capabilities. Among these applications we mention targeted advertising [1], information retrieval and web search [2], automatic tagging and linking [3], and text categorization [4]. The importance of SR is particularly evident when attempting to categorize short texts (e.g., ads, tweets, search queries) where standard bag-of-words representation is not sufficiently effective [5].

The goal in SR is to quantify the intensity of how much two target terms are related to each other, and all relation types between these terms might be considered. These relations can be among the linguistically formal ones, which typically have a name (e.g., synonyms, hypernyms, etc.), but in general, such relations can be informal in the sense that they have not been coined, and they express some (perhaps complex) association between the two terms. For example, consider the following term pairs: (`Michael Jordan,Basketball`), (`Madonna,Pop`), and (`Marilyn Monroe,Movie`). All three pairs, (`X,Y`), are strongly related via a common relation.[1] Thus, the SR task involves all possible relations whose number is in principle unbounded. We note that in a restricted version of SR, called *semantic similarity*, one considers only synonymy relations. As argued by Budanitsky and Hirst [6] and others, SR is considered more general than semantic similarity, and in this sense it is more difficult.

In this work we consider the SR task and we aim to correctly qualify the relatedness of two given terms where the underlying relation can be linguistic or informal. However,

[1] The relation we had in mind is "X is an all times Y star"; other variations are sensible.

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 744–759, 2012.

we do not aim at identifying or characterizing the underlying relation. Note also that in the standard SR setting we consider here (see definitions in Section 3), the terms to be evaluated for relatedness are provided without a context, unlike typical disambiguation tasks. Hence, as most existing works on SR focusing on this or equivalent setup, we do not aim at directly solving the disambiguation problem along the way.

SR is an elusive concept. While a rigorous mathematical definition of SR is currently beyond grasp, the concept is intuitively clear to everyone. Indeed, the popular conception of SR is reflected in its nebulous Wikipedia entry (at the moment of writing this) stating that: *In essence, semantic similarity, semantic distance, and semantic relatedness all mean, "How much does term A have to do with term B"?* Clearly, SR "has to do" with the understanding of meaning – a grand challenge in AI research. Can a computer program quantify the extent to which two terms share the same meaning?

The statistical NLP and AI communities have adopted a pragmatic modus operandi to these questions: even if we don't know how to define SR, we can still create computer programs that generate useful SR assesments. Indeed, a number of effective heuristic approaches to SR have been proposed, and this line of work has proven to be rewarding, see e.g., [7, 8]. In particular, it has been shown that useful SR scores can be systematically extracted from large lexical databases or electronic repositories of common-sense and domain-specific background knowledge.

With the exception of a few papers, most of the algorithms proposed for SR valuation have been following an *unsupervised* learning or *knowledge engineering* procedures. Such SR valuation functions have been generated, for the most part, using some handcrafted formula applied to semantic information that is extracted from a (structured) background knowledge corpus. The proposed methods have employed a number of interesting techniques, some of which are discussed in Section 2.

One motivation for the present work is the realization that SR assessments are *subjective* and often *relative*, rather than objective and absolute. While we can expect some kind of consensus among people on the (relative) relatedness valuations of basic terms, the relatedness assessments of most terms depend on many subjective factors such as literacy, intelligence, context, time and location. For example, the name `Michael Jordan` is generally strongly related to `Basketball`, but some people in the machine learning community may consider it more related to `Machine Learning`. As another example, consider WordSim353 [9], a standard benchmark dataset for evaluating and comparing SR measures. This benchmark contains some controversial relative preferences between word pairs such as (`Arafat`, `Peace`) vs. (`Arafat`, `Terror`) and (`Jerusalem`, `Israel`) vs. (`Jerusalem`, `Palestinian`). Can you tell which pair is more related in each instance? Obviously, the answer must be personal/subjective.

This sensitivity of SR to subjective factors should make it very hard, if not impossible, to satisfy all SR needs using a single "universal" method. Indeed, some published SR measures outperform others in certain benchmarks tests and underperform in others. For example, Strube and Ponzetto [10] mentioned that the WordNet-based measures perform better than the Wikipedia-based measures on the Rubenstein and Goodenough benchmark, but the WordNet methods are inferior over WordSim353.

In this work we propose a novel *supervised* approach to learning SR from examples. Following Agirre at al. [11] we model SR learning as a binary classification problem

where each instance encodes the relative relatedness of two term pairs. Given a labeled training set our goal is to learn an SR function capable of determining the labels of unobserved instances. We present an empirical risk minimization (ERM) algorithm that learns by inducing a weighted measure of terms co-occurrence defined over a background knowledge corpus of free-text documents. The labeled examples are used to fit this model to the training data. The resulting algorithm is relatively simple, has only few hyper-parameters, and is corpus independent. Our experiments show that the algorithm achieves notable generalization performance. This is observed over a wide range of experiments on a number of benchmarks. We examine and demonstrate the effectiveness of our algorithm using two radically different corpora: an old version of Wikipedia and the books in the Project Gutenberg.

2 Related Work

SR techniques typically rely on some kind of world or expert knowledge, which we term here *background knowledge (BK)* corpus. The BK corpus is a key element in many methods and we categorize existing SR techniques into three main families according to type and structure of their BK corpus. *Lexical* methods rely on lexical databases such as WordNet or Rodget's Thesaurus. *Wiki* methods rely on structured BK corpora like Wikipedia or the Open Directory Project (DMOZ). Finally, SR techniques that rely on unstructured text collections are referred to as *structure-free* methods. Currently, the largest publicly available SR benchmark dataset is WordSim353 [9]. Encompassing a variety of semantic relations, WordSim353 has been providing a focal point to empirical SR research in the past years. In the semantic relatedness literature it is common to evaluate relatedness ranking using the Spearman correlation. Hence, in the sequel we mention WordSim353 Spearman correlation scores in cases where they were reported.

Unsupervised Methods: Lexical, Wiki and Structure-Free. Numerous *lexical methods* utilize the WordNet database, which organizes words in *synsets* (sets of synonyms). The lexical relations among synsets are categorized into types such as synonyms, antonyms and hypernyms. Another lexical database is the well-known Roget's Thesaurus. Similarly to WordNet, Rodget's Thesaurus contains groups of terms, called *semicolon groups*. These groups are inter-linked, but the links are not lexically annotated as in WordNet. Lexical SR methods typically view the lexical database as a graph whose nodes are terms and edges are lexical relations.

Several researchers have defined SR measures combining lexical links with structure free corpora [12–14]. Others only utilize the lexical links [15–17]. Yet others [18–20] use WordNet by taking advantage of *glosses* (terms' definitions). Refer to [6, 21, 22] for various other lexical methods.

Structure in Wiki BK corpora can be manifested in various ways, and the most important ones are semantic coherency of documents and titles [8, 23], meaningful interlinks [24, 25], and hierarchical categorization [26]. Many such *Wiki methods* associate an article in Wikipedia to each term and utilize known or newly proposed measures between Wikipedia's articles. Examples include Strube and Ponzetto [10], as well as Gabrilovich and Markovitch's celebrated Explicit Semantic Analysis (*ESA*) method [8],

whereby each term is represented as a sparse vector containing a non-zero component for each significant Wikipedia article in which it appears, and the SR score of two terms is defined as the cosine of their vectors. ESA achieved a correlation of 0.75 with WordSim353, and is used as a subroutine in applications [23, 25, 27]. The Temporal Semantic Analysis (*TSA*) measure proposed by Radinsky et al. [23] recently achieved 0.82 correlation for WordSim353, which is the best known *unsupervised* result on WordSim353.

Structure-free methods posit that SR is a function of the co-occurrence statistics of a pair of terms in a BK corpus. Lin [28] proposed information-based methods to define and quantify term similarity. Dagan et al. [29] and Terra and Clarke [30] experimented with various statistical co-occurrence measures for estimating SR from structure-free corpora. Deerwester et al. [31] used algebraic representation of the BK. Applying this measure, Finkelstein et al. [9] achieved 0.56 correlation with WordSim353. Reisinger and Mooney [32] obtained a correlation of 0.77 with WordSim353 by generating for a term t a feature vector for each context in which t appears.

Supervised Methods. There have been a few successful attempts to utilize *supervised* learning techniques for constructing SR functions. For the most part, these works follow a similar methodology whereby the features of a learning instance are assembled from scores obtained from various unsupervised methods (such as those discussed above). Using this feature generation approach, the existing works then resorted to known inductive learning schemes, such as support vector machines (SVMs). For a partial list of results, refer to [10, 11, 27, 33]. Haralambous and Klyuev reported on 0.8654 correlation score with WirdSim353 [27], which is the best that was ever reported. The second best result is by Agirre et al. [11], achieving 0.78 correlation with WordSim353. These two works are also the closest to ours, mainly in their formulation of the learning problem. However, our solution methodology is fundamentally different.

3 Problem Setup

We consider a fixed corpus, $\mathcal{C} \triangleq \{c_1, c_2, \ldots, c_N\}$, defined to be a set of contexts. Each *context* c_i, $i = 1, \ldots, N$, is a textual unit conveying some information in free text. In this work we consider contexts that are sentences, paragraphs or whole documents. Let $D \triangleq \{t_1, t_2, \ldots, t_d\}$ be a *dictionary* consisting of a desired subset of all the terms appearing in the corpus. A term may be any frequent phrase (unigram, bigram, trigram, etc.) in the corpus, e.g., "book", "New York", "The Holly Land." Ultimately, our goal is to automatically construct a function $f(t_1, t_2)$ that correctly ranks the relatedness of the terms $t_1, t_2 \in D$ in accordance with the subjective semantics of a given user. We emphasize that we do not require f to provide absolute scores but rather a relative value inducing a complete order over the relatedness of all terms. In reality this total order assumption doesn't hold, since the comparison between two term pairs not sharing any term might be meaningless. Furthermore, human preferences may contain cycles, perhaps due to comparisons made using different features (as in the rock-paper-scissors game).

Our goal is to construct the function f using supervised learning. Specifically, the user will be presented with a training set $\{X_1, \ldots, X_m\}$ to be labeled, where each $X_i \triangleq (\{t_1^i, t_2^i\}, \{t_3^i, t_4^i\})$ is a quadruple of terms. The binary label, $y_i \in \{\pm 1\}$, of

the instance X_i should be $+1$ if the terms in the first pair $\{t_1^i, t_2^i\}$ are more related to each other than the terms in the second pair $\{t_3^i, t_4^i\}$, and -1 otherwise. Each quadruple along with its label, (X_i, y_i) is also called a *preference*.

Denote by $S_m \triangleq \{(X_1, y_1), \ldots, (X_m, y_m)\}$, a set of labeled training examples received from the user. We assume that if $(X, y) \in S_m$ then $(X, -y) \notin S_m$. A binary classifier in our context is a function $h : D^4 \to \{\pm 1\}$ satisfying, for all $(\{t_1, t_2\}, \{t_3, t_4\}) \in D^4$, the "anti-symmetry" condition $h(\{t_1, t_2\}, \{t_3, t_4\}) = -h(\{t_3, t_4\}, \{t_1, t_2\})$. The 0/1 *training error* of h is, $R_m(h) \triangleq \frac{1}{m} \sum_i \mathbb{I}\{h(X_i) \neq y_i\}$. The standard underlying assumption in supervised learning is that (labeled) instances are drawn i.i.d. from some unknown distribution $P(X, Y)$ defined over $D^4 \times \{\pm 1\}$. The classifier h is chosen from some hypothesis class $\mathcal{H}$. In this work we focus on the *realizable setting* whereby labels are defined by some unknown *target hypothesis* $h^* \in \mathcal{H}$. Thus, the underlying distribution reduces to $P(X)$. The performance of a classifier h is quantified by its true or (0/1) *test error*, $R(h) \triangleq \mathbf{E}_P\{h(X) \neq h^*(X)\}$.

Why do we choose to ask the user about pairwise preferences rather than requesting an absolute relatedness score of a single pair of terms? Our choice is strongly motivated by recent work showing that answers to such questions are more accurate than answers to questions about absolute quality. In order to extract an absolute score, a user must rely on some implicit global scale, which may or may not exist. For a sample of literature justifying this general approach (both theoretically and empirically) see [34, 35].

4 Adaptive Measure

Recognizing the widely accepted idea that the SR of two terms is a function of their co-occurrence pattern in documents, we would like to somehow measure co-occurrence using a BK corpus where such patterns are manifested. Therefore, a major component of the proposed algorithm is an appropriate co-occurrence measure. However, we also require adaptivity to specific user's subjective relatedness preferences. Our observation is that such adaptivity can be achieved by learning from examples user specific weights to be assigned to contexts. Overall, our approach is to construct a reasonable initial model, derived only from the BK corpus (without supervision), which fits a rough general consensus on relatedness of basic terms. This initial model is the starting point of a learning process that will refine the model to fit specific user preferences.

We examined various co-occurrence indices, such as Jaccard measure, pointwise mutual information, KL- and Jensen-Shannon divergences, and latent semantic analysis. Based on this study and some other results [29, 30, 36], we selected the normalized semantic distance measure of Cilibrasi and Vitanyi [37].[2] Indeed, this measure by itself can achieve a high 0.745 Spearman correlation with WordSim353 (via our implementation using Wikipedia as the BK corpus) thus providing a very effective starting point. We note that information measures are also effective, but not quite as good.[3] We also find it appealing that this measure was derived from solid algorithmic complexity principles.

[2] Note that Cilibrasi and Vitanyi termed this function "Google similarity distance" and applied it by relying on Google to retrieve proxies for co-occurrence statistics. In our discussion co-occurrence statistics can be obtained in any desirable manner.

[3] Pointwise mutual information achieved correlation of 0.73 with WordSim353[36].

Cilibrasi and Vitanyi defined the *semantics* $S(t_1, \ldots, t_n)$ of the terms $t_1, \ldots, t_n$, as the set of all contexts in which they appear together. Than they defined the *normalized semantic distance (*NSD*)* between t_1, t_2 to be

$$\mathsf{NSD}\,(t_1, t_2) \triangleq \frac{\max\{\log(|S(t_1)|), \log(|S(t_2)|)\} - \log(|S(t_1, t_2)|)}{\log(Z) - \min\{\log(|S(t_1)|), \log(|S(t_2)|)\}},$$

where $Z \triangleq \sum_{t_1, t_2 \in D} |S(t_1, t_2)|$. The NSD function, like any other absolute scoring function for pairs, induces a permutation over all the term pairs, and therefore, can be utilized as a classifier for SR preferences, as required. However, this classifier is constructed blindly without any consideration of the user's subjective preferences. To incorporate user subjective preferences we introduce a novel extension of NSD that allows for assigning weights to contexts. Define the *weighted semantics* $WS(t_1, \ldots, t_n)$ of the terms $t_1, \ldots, t_n$ as $WS(t_1, \ldots, t_n) \triangleq \sum_{c \in S(t_1, \ldots, t_n)} w(c)$, where $w(c) \in \mathbb{R}^+$ is a weight assigned to the context c, where we impose the normalization constraint $\sum_{c \in \mathcal{C}} w(c) = |\mathcal{C}| = N$. Thus, given a BK corpus, $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$, and a set W of weights, $W \triangleq \{w(c_1), w(c_2), \ldots, w(c_N)\}$, we define *weighted normalized semantic distance (*WNSD*)* between t_1 and t_2 is,

$$\mathsf{WNSD}_W(t_1, t_2) \triangleq \frac{\max\{\log(WS(t_1)), \log(WS(t_2))\} - \log(WS(t_1, t_2))}{\log(Z) - \min\{\log(WS(t_1)), \log(WS(t_2))\}},$$

where Z is a normalization constant, $Z \triangleq \sum_{t_1, t_2 \in D} WS(t_1, t_2)$. We call the set W of weights a *semantic model*. Our goal is to learn a good model from labeled examples.

Recall that our objective is to quantify the relatedness of two terms regardless of the types of relations that link these terms. Is it really possible to learn a single model W that will encode coherent semantics universally for all terms and all relations?

At the outset, this objective might appear hard or even impossible to achieve. Additional special obstacle is the modeling of synonym relations. The common wisdom is that synonym terms, which exhibit a very high degree of relatedness, are unlikely to occur in the same context (see, e.g., [6, 28]), especially if the context unit is very small (e.g., a sentence). Can our model capture also similarity relations? We empirically investigate this issue and answer this question in the affirmative.

5 The SemanticSort Algorithm

Let $f_W : D \times D \to \mathbb{R}^+$ be any adaptive co-occurrence measure satisfying: (i) each context has an associated weight in W; (ii) $f_W(t_1, t_2)$ monotonically increases with increasing weight(s) of context(s) in $S(t_1, t_2)$; and (iii) $f_W(t_1, t_2)$ monotonically decreases with (increasing) weight(s) of context(s) in $S(t_1) \setminus S(t_1, t_2)$ or $S(t_2) \setminus S(t_1, t_2)$.

We now present a learning algorithm that can utilize any such function. We later apply this algorithm while instantiating this function to WNSD, which clearly satisfies the required properties. Note, however, that many known co-occurrence measures can be extended (to include weights) and be applied as well.

Relying on f_W we would like utilize empirical risk minimization (ERM) to learn an appropriate model W of context weights so as to be consistent with the training set S_m.

Algorithm 1. SemanticSort$(S_m, \alpha, \alpha_{max}, \epsilon, \lambda)$

Initialize:
$W \leftarrow \overrightarrow{1}$, $\Delta_{prev} \leftarrow MaxDoubleValue$
repeat
  $\Delta \leftarrow 0$
  For all $e = ((\{t_1, t_2\}, \{t_3, t_4\}), y) \in S_m$ **do**
    If $(y == -1)$ **then**
      $(\{t_1, t_2\}, \{t_3, t_4\}) \leftarrow (\{t_3, t_4\}, \{t_1, t_2\})$
    $score_{12} \leftarrow f_W(t_1, t_2)$ $score_{34} \leftarrow f_W(t_3, t_4)$
    If $(score_{12} < score_{34})$ **then**
      {This is an unsatisfied example.}
      $\lambda_{up} \leftarrow \frac{\alpha \cdot \lambda(\Delta_e)+1}{\alpha \cdot \lambda(\Delta_e)}$, $\lambda_{dn} \leftarrow \frac{1}{\lambda_{up}}$
      $\Delta \leftarrow \Delta + \Delta_e$
      for all $c \in S(t_1, t_2)$ **do** $w(c) \leftarrow w(c) \cdot \lambda_{up}$
      for all $c \in S(t_3, t_4)$ **do** $w(c) \leftarrow w(c) \cdot \lambda_{dn}$
      Normalize weights s.t. $\sum_{c \in \mathcal{C}} w(c) = |\mathcal{C}|$
  If $(\Delta - \Delta_{prev} + \epsilon \geq 0)$ **then**
    $\alpha \leftarrow 2 \cdot \alpha$
    If $(\alpha \geq \alpha_{max})$ **then Return**
  $\Delta_{prev} \leftarrow \Delta$
until $\Delta == 0$

To this end we designed SemanticSort, an algorithm that minimizes the training error over S_m by fitting appropriate weights to f_W. A pseudocode is provided in Algorithm 1.

The inputs to SemanticSort are S_m, a learning rate factor α, a learning rate factor threshold α_{max}, a decrease threshold ϵ, and a learning rate function λ. When a training example is not satisfied, e.g., $e = (X = (\{t_1, t_2\}, \{t_3, t_4\}), y = +1)$ and $f_W(t_1, t_2) < f_W(t_3, t_4)$, we would like to increase the semantic relatedness score of t_1 and t_2 and decrease the semantic relatedness score of t_3 and t_4. SemanticSort achieves this by multiplicatively promoting/demoting the weights of the "good"/"bad" contexts in which t_1, t_2 and t_3, t_4 co-occur. The weight increase (resp., decrease) depends on λ_{up} (resp., λ_{dn}), which are defined as follows. $\lambda_{up} \triangleq \frac{\alpha \cdot \lambda(\Delta_e)+1}{\alpha \cdot \lambda(\Delta_e)}$, $\lambda_{dn} \triangleq \frac{1}{\lambda_{up}}$.

SemanticSort uses λ to update context weights in accordance with the error magnitude incurred for example e, defined as $\Delta_e \triangleq |f_W(t_1, t_2) - f_W(t_3, t_4)|$. Thus, we require that λ is a monotonically decreasing function so that the greater Δ_e is, the more aggressive λ_{up} and λ_{dn} will be. The learning speed of the algorithm depends on these rates, and overly aggressive rates might prevent convergence due to oscillating semantic relatedness scores. Hence, SemanticSort gradually refines the learning rates as follows. Define $\Delta \triangleq \sum_{e \text{ is not satisfied}} \Delta_e$, as the total sum of the differences over unsatisfied examples. We observe that if Δ decreases at least in ϵ in each iteration, then SemanticSort converges and the learning rates remain the same. Otherwise, SemanticSort will update the learning rate to be less aggressive by doubling α. Therefore, we require that $0 < \epsilon$. Note that the decrease of Δ is only used to control convergence, but we test SemanticSort using the 0/1 loss function as described in Section 3. SemanticSort iterates over the examples until its hypothesis satisfies all of them, or α exceeds α_{max}. Thus, empirical risk minimization in our context is achieved by minimizing Δ.

6 Empirical Evaluation

To evaluate the effectiveness of SemanticSort we conducted several experiments. One of the barriers in designing these experiments is the lack of labeled dataset of term quadruples as required by our model. The common benchmark datasets are attractive because they were labeled by human annotators, but these datasets are rather small. When considering a small real world application involving even 500 vocabulary terms, we need to be able to compare the relatedness of many of the $\binom{500}{2} = 124,750$ involved pairs. However, the largest available dataset, WordSim353, contains only 353 pairs. Although we utilized all available datasets in our experiments (see below), we sought a benchmark of significantly larger scale in order to approach real world scenarios.

Gutenberg Semantic Score (GSS). Without access to a humanly annotated dataset of a large scale, we synthesized a labeled dataset as follows. Noting that a vocabulary of 1000-2000 words covers about 72%-80% of written English texts [38], we can envision practical applications involving vocabularies of such sizes. We therefore selected a dictionary D_n consisting of the n most frequent English words ($n = 500, 1000$). For each of the $\binom{n}{2}$ term pairs over D_n we used an independent corpus of English texts, namely the Gutenberg Project, to define the SR score of pairs, using the NSD method, applied with sentence based contexts. We refer to this as the *Gutenberg Semantics Score (GSS)*.

Project Gutenberg is a growing repository that gathers many high quality and classic literature that is freely available on the web. For example, among the books one can find `Alice's Adventures in Wonderland`, `The Art of War`, `The Time Machine`, `Gulliver's Travels`, and many well known fiction ebooks. Currently, Project Gutenberg offers over 36,000 ebooks.[4]

While GSS is certainly not as reliable as human generated score (for the purpose of predicting human scores), it is positively correlated with human annotation, achieving 0.58 Spearman correlation with the WordSim353 benchmark. Given a set of term pairs together with their SR scores (such as those generated by GSS), we construct a labeled set of preferences according to SR scores (see definitions in Section 3).

We emphasize that the texts of the Project Gutenberg were taken conclusively and as is, without any modifications, to avoid any selection bias.[5] Nevertheless, despite its statistical correlation to human annotation, our main objective isn't to evaluate absolute performance scores, but rather to see if generalization can be accomplished at this scale while using an extremely small fraction of the available training examples.

Background Knowledge Corpora. An integral part of the SemanticSort model is its BK corpus. We conducted experiments using two corpora. The first corpus is the snapshot of Wikipedia from 05/11/05 preprocessed using Wikiprep.[6] Following [8], in order to remove small and overly specific articles, we filtered out articles containing either

[4] In this work we used a complete older version of Project Gutenberg from February 1999 containing only 1533 texts bundled by Walnut Creek CDROM.

[5] The GSS dataset is available at `http://www.cs.technion.ac.il/~rani/semantic_relatedness`.

[6] Wikiprep is an XML preprocessor for Wikipedia, available at `http://www.cs.technion.ac.il/~gabr/resources/code/wikiprep`.

less than 100 non-stopword terms and/or less than 5 incoming links and/or less than 5 outgoing links. The second corpus we used is the Project Gutenberg mentioned above. We emphasize that in all experiments involving GSS scores only Wikipedia was used as the BK corpus. Also, in each experiment we either used Wikipedia or Gutenberg as a BK corpus and not both. In all the experiments we ignored stopwords and stemmed the terms using Porter's stemmer. Finally, We considered three types of contexts: sentences, paragraphs and whole documents. Sentences are parsed using '.' as a separator without any further syntax considerations; paragraphs are parsed using an empty line as a separator. No other preprocessing, filtering or optimizations were conducted.

Evaluation Methodology. Consider a collection P of preferences, where each preference is a quadruple, as define in Section 3. When we evaluate performance of the algorithm w.r.t. a training set of size m, we choose an m-subset, $S_m \subseteq P$ uniformly at random. The rest of the preferences in $P \setminus S_m$ are taken as the test set.[7] However, if $P \setminus S_m$ remains very large, only 1,000,000 preferences, chosen uniformly at random from $P \setminus S_m$, are taken for testing. The training set S_m is fed to SemanticSort, which generates an hypothesis h, consisting of a weights vector W that includes a component for each context in $\mathcal{C}$. Then we apply the hypothesis on the test set and calculate the resulting accuracy (using the 0/1 loss function). This quantity provides a relatively accurate estimate of (one minus) the true error $R(h)$. In order to obtain a learning curve we repeat this evaluation procedure for a monotonically increasing sequence of training set sizes. The popular performance measure in SR research is the Spearman correlation coefficient of the ranking obtained by the method to the ground truth ranking. Therefore, we also calculated and reported it as well. In addition, in order to gain statistical confidence in our results, we repeated the experiments for each training set size multiple times and reported the average results. For each estimated average quantity along the leaning curve we also calculated its standard error of the mean (SEM), and depicted the resulting SEM values as error bars.

Experiment 1: Large Scale. In order to evaluate SemanticSort on ambitious, large scale and quite realistic scenario, we conducted the following experiments. Taking D_{1000} (the top 1000 most frequent words in Wikipedia) we considered all possible preferences. Note that the number of preferences associated with D_{1000} is huge, containing about $10^{12}/4$ quadruples. We labeled the preferences according to GSS as described above. In generating the learning curve we were only able to reach $m = 2,000,000$ training examples, thus utilizing an extremely small fraction of the available preferences (the largest fraction is about 10^{-5}). Figure 1 presents 0/1 test accuracy and Spearman correlation learning curves. On this figure we also mark the results obtained by two unsupervised methods: (i) NSD using Wikipedia as BK corpus with paragraph level contexts; (ii) the well known ESA method using the same filtered Wikipedia snapshot mentioned above. Both these unsupervised performance scores were calculated by us using our implementations of these methods. It is evident that SemanticSort successfully generalized the training sample and accomplished a notable improvement

[7] Formally speaking, this type of sampling without replacement of the training set, is within a standard transductive learning model [39, Sec. 8.1,Setting 1] .

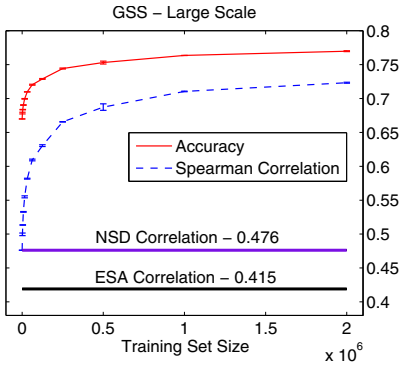


Fig. 1. Experiment 1 (large scale) - Learning curves for test accuracy (solid) and test correlation (dashed), with standard error bars

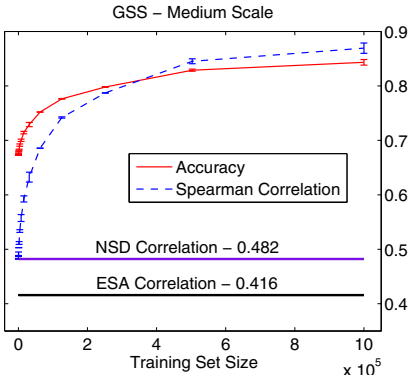


Fig. 2. Experiment 2 (medium scale) - Learning curves for test accuracy (solid) and test correlation (dashed),with standard error bars

over its starting point. We believe that these results can serve as a proof of concept and confirm SemanticSort's ability to handle real world challenges.

Experiment 2: Medium Scale. We repeated the previous experiment, now with D_{500}, taken to be a random of D_{1000}. All other experiment parameters were precisely as in Experiment 1. The resulting learning curves are shown in Figure 2. Clearly, this medium scale problem gave rise to significantly higher absolute performance scores. We believe that the main reason for this improvement (over the large scale experiment) is merely the use of a larger fraction of preferences in training.

Experiment 3: Small Scale. As mentioned in Section 2, many of the known techniques, including the published supervised methods, evaluated performance with respect to the WordSim353 benchmark. In order to link the proposed approach to the current literature we also conducted an experiment using WordSim353 as a source for labeled preferences. This experiment serves three purposes. First, it can be viewed as a sanity check for our method, now challenging it with humanly annotated scores. Second, it is interesting to examine the performance advantage of our supervised approach vs. no systematic supervision as obtained by the unsupervised methods (we already observed in Experiments 1&2 that our supervised method can improve the scores obtained by ESA and NSD). Finally, using this experiment we are able compare between SemanticSort and the other known supervised methods that so far have been relying on SVMs.

Figure 3 shows the learning curves obtained by SemanticSort applied with paragraph contexts using either Wikipedia or Gutenberg (but not both together) as a BK corpus. The lower horizontal line, at the 0.82 level, marks the best known *unsupervised* result obtained for WordSim353 [23]. The upper horizontal line, at the 0.8654 level, marks the best known *supervised* result [27]. It is evident that quite rapid learning is accomplished using either the Wikipedia or the Gutenberg models, but Wikipedia

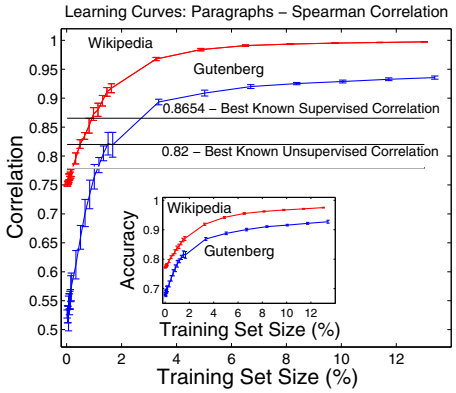


Fig. 3. Experiment 3 (small scale) - Learning curves for test correlation and test accuracy (sub-plot) with standard error bars using either Wikipedia or Gutenberg.

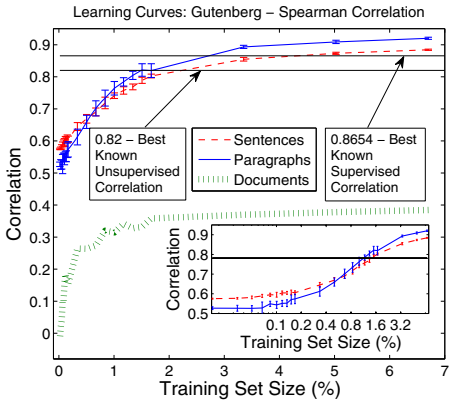


Fig. 4. Experiment 3 (small scale) - Learning curves for test correlation with standard error bars using Project Gutenberg. The sub-plot uses logarithmically scaled X-axis.

enables significantly faster learning and smaller sample complexity for each error level. The curves in the internal panel show the corresponding test *accuracies* (0/1 loss) for the same experiments. Note that meaningful comparisons between SemanticSort and the other (SVM based) supervised methods (described in Section 2) can only be made when considering the same train/test partition sizes. Unlike our experimental setting, both Agirre et al. [11], and Haralambous and Klyuev [27] achieved their reported results (0.78 and 0.8654 correlation with WordSim353, respectively) using 10-fold cross validation, thus utilizing 90% of the available labeled preferences for training. When considering only the best results obtained at the top of the learning curve, SemanticSort outperforms the best reported supervised performance after consuming 1.5% of all the available WordSim353 preferences using the Wikipedia model and after consuming 3% of the preferences using the Project Gutenberg model.

Figure 4 depicts three Gutenberg learning curves: one for each context type. The internal panel zooms into the same curves of sentence- and paragraph-based contexts, now with logarithmically scaled X-axis to emphasize their differences. As before, the lower (resp., upper) horizontal line at 0.82 (resp., 0.8654) marks the best known unsupervised (resp., supervised) result for WordSim353 [23] (resp., [27]). Clearly, paragraph contexts exhibit the best test performance for almost all training set sizes. In contrast, contexts consisting of whole documents perform poorly, to the extent that even after utilizing the largest training set size, they are still way behind sentences and paragraphs (even without using a single labeled example). A similar comparison (not presented) for Wikipedia contexts showed entirely different picture with all contexts exhibiting very similar (and almost indistinguishable) performance as shown for paragraphs in Figure 3.

Experiment 5: Semantic Similarity. Synonymous relations are considered among the most prominent semantic relations. *Semantic similarity* is a sub-domain of SR where one attempts to assess the strength of synonymous relations. A widely accepted

approach to handle synonyms (and antonyms) is via distributional similarity [6, 28]. In this approach, to determine the similarity of terms t_1 and t_2 we consider $D(t_1)$ and $D(t_2)$, the "typical" distributions of terms in close proximity to t_1 and t_2, respectively. It is well known that these distributions tend to resemble whenever t_1 is similar to t_2, and vice versa. In contrast, SemanticSort computes its similarity scores based on co-occurrence counts, and the conventional wisdom is that synonyms tend not to co-occur. Can we expect SemanticSort to handle synonymous relations?

We examine and analyze the behavior of a paragraph-based SemanticSort on a specialized semantic similarity task. To this end, we use the semantic similarity datasets, namely Rubenstein and Goodenough (R&G) [40] and Miller and Charles (M&C) [41].

Figure 5 depicts the results obtained for the M&C dataset. The upper (resp. lower) horizontal line, at the 0.92 (resp. 0.9) level, marks the best known *supervised* (resp. *unsupervised*) results obtained for the M&C dataset [11] (resp. [14, 42]). Figure 6 depicts the results obtained for the R&G dataset. The upper (resp. lower) horizontal line, at the 0.96 (resp. 0.8614) level, marks the best known *supervised* (resp. *unsupervised*) results obtained for R&G dataset [11] (resp. [17]). The learning curves depicted in both figures clearly indicate that learning synonyms using our method is an achievable task, and in fact, can improve upon the distributional similarity methods. While synonyms and antonyms co-occur infrequently, they still do co-occur. It is a nice property of our model that it can leverage these sparse co-occurrence counts and accurately detect synonyms by sufficiently increasing the weights of their mutual contexts.

7 Model Interpretability

The model learned by SemanticSort is encoded in its weight vector W. In this section we summarize our preliminary study to explore the model W and gain some insight into its structure. Are the weights in W "arbitrarily" optimized to reduce the training error, or is it the case that they are organized in a meaningful and interpretable manner? Can we learn from W something about the human rater(s) who tagged the training set? Can something on their world knowledge and/or intellectual interests be inferred?

Trying to answer the above questions we conducted the following study. The results fall short of fully answering these questions, but they are indicative and suggest that the model W contains useful information that perhaps could be utilized in applications. In our experiments, due to the absence of human annotating resources, we again synthesized "human" raters whose knowledge is focused on specific topics.

Given a specific topic T in Wikipedia (e.g., sports) we extracted the set S_T of documents pertaining to T (using the Wikipedia topic tags), and partitioned S_T uniformly at random into two subsets, S_T^1 and S_T^2. We used S_T^1 for labeling, and S_T^2, as part of the BK corpus together with the rest of the Wikipedia corpus. Our synthetic rater annotated preferences based on NSD applied over S_T^1, whose articles were partitioned to paragraph units. We call the resulting semantic preferences the T-*semantics*.

Taking D_{1000} as a dictionary, we generated a training set by sampling uniformly at random $m = 2,000,000$ preferences, which were tagged using the T-semantics. We then applied SemanticSort to learn the T-semantics using this training set while utilizing S_T^2 (as well as the rest of Wikipedia) as a BK corpus, whose documents were parsed to the paragraph level as well. We then examined the resulting W_T model.

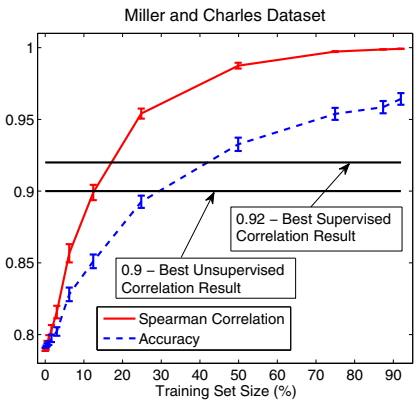


Fig. 5. Experiment 5 – Semantic similarity with Miller & Charles dataset

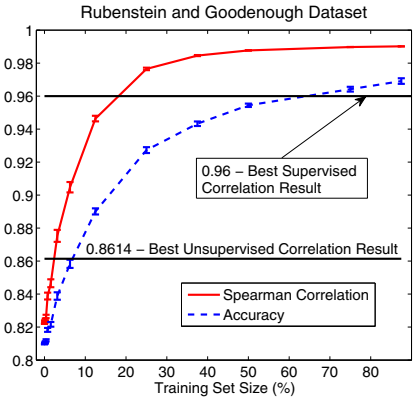


Fig. 6. Experiment 5 – Semantic similarity with Rubenstein and Goodenough dataset

Table 1. Model Interpretability - Top 10 related terms according to Music and Sports Semantics

#	play		player		record		club	
	Music	Sports	Music	Sports	Music	Sports	Music	Sports
1	band	game	instrument	play	release	set	dance	football
2	guitar	team	play	league	album	season	night	league
3	instrument	season	replace	game	label	win	heart	cup
4	perform	player	join	season	band	career	fan	play
5	time	football	guitar	born	song	finish	local	divis
6	year	first	technique	team	first	run	house	season
7	role	score	key	football	new	game	London	manage
8	tour	club	example	professional	studio	won	scene	success
9	two	year	football	baseball	production	score	mix	found
10	new	career	hand	major	sign	second	radio	player

Two topics T were considered: Music and Sports, resulting in two models: W_{music} and W_{sports}. In order to observe and understand the differences between these two models, we identified and selected, before the experiment, a few target terms that have ambiguous meanings with respect to Music and Sports. The target terms are: `play`, `player`, `record`, `club`. Table 1 exhibits the top 10 most related terms to each of the target terms according to either W_{music} or W_{sports}. It is evident that the semantics portrayed by these lists are quite different and nicely represent their topics as we may intuitively expect. The table also emphasizes the inherent subjectivity in SR analysis, that should be accounted for when generating semantic models.

Given a topical category C in Wikipedia, and a hypothesis h, we define the *aggregate C-weight according to h*, to be the sum of the weights of all contexts that belong to an article that is categorized into C or its Wikipedia sub-categories. Also, given a topic T, we denote by h^T_{init}, its initial hypothesis and by h^T_{final}, its final hypothesis (after learning). In order to evaluate the influence of the labeling semantics on h^T_{final}, we

calculated, for each topic T, the difference between its aggregate C-weights according to h^T_{init}, and according to h^T_{final}.

Figure 7 presents the increase/decrease in those aggregate C-weights for some of Wikipedia's major categories C. In both cases of labeling topics, Music or Sports, observe that by and large the aggregate weights of categories that are related to the labeling topic were increased, while weights of unrelated categories were decreased. Quite surprisingly, when considering the Music topic, many mathematical categories dramatically increased their weight. Various other interesting relations are highlighted by this process. For example, notice the sharp decrease of political topics in the Sports model (right), and the decrease of wars and military in the music model (left).

While these results aren't conclusive (and can certainly be viewed as anecdotal), we believe they do indicate that the weights in W are organized in a meaningful and interpretable manner, which encodes the labeling semantics as a particular weight distribution over the corpus topics. In addition, not only did SemanticSort identified the labeler BK, it also revealed some other unexpected topical relations.

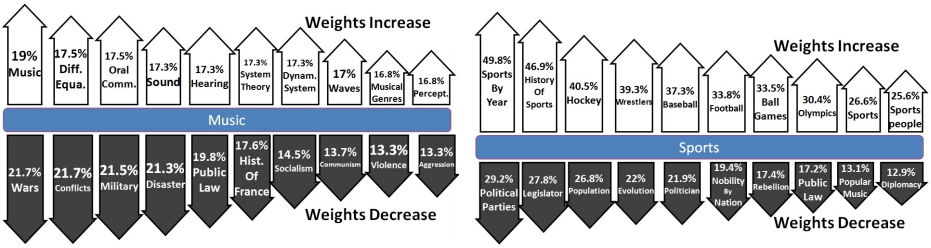


Fig. 7. Model Interpretability – Weights increase (upper/white) and decrease (lower/black) of Wikipedia's major categories according to Music hypotheses (left) and Sports (right)

8 Concluding Remarks

Building on successful and interesting ideas, we presented in this work a novel supervised method for learning SR. The proposed SemanticSort algorithm exhibits interesting performance over a large and medium scale problems and competitive performance on small scale problems. The SemanticSort algorithm performs feature re-weighting as done by multiplicative update algorithms such as Winnow. Moreover, any (kernel) inner product algorithm such as SVM can be applied on a vectorial representation of features corresponding to contexts as induced by the proposed semantic model. However, from a learning theoretic perspective, the use of general purpose classification algorithms over those features can be problematic due to the huge number of contexts in our model, which may unfortunately lead to overfitting. In contrast, by its construction with appropriate f_W function (such as the WNSD), SemanticSort can only output permutations over the set of word pairs of size $\binom{|D|}{2}$. Thus, the hypothesis space considered by SemanticSort is a subset of those permutations, whose VC-dimension can be shown to be precisely $\binom{|D|}{2} - 1$. Moreover, the utilization of the BK corpus through the WNSD measure, provides further capacity reductions by placing many constraints

on the set of allowable permutations. For example, observe that SemanticSort only updates the weights of contexts that include both terms in a given pair, so it cannot change the semantic relatedness score of terms that do not co-occur.

It would be nice to gain further theoretical insights on SemanticSort that can explain its successful performance. Another interesting question is how to extend our SR model and the algorithm to allow for active learning. The performance of SemanticSort depends on appropriate choice of the learning rate function $\lambda(\cdot)$. It is desired to explore possibilities for automatically adapting this learning rate. Our results indicate that high quality SR can be learned with markedly different types of BK corpora. It would be very interesting to obtain better understanding on the role of the BK corpus, and perhaps even quantify the effectiveness of a given BK corpus. Finally, SemanticSort is a general ranking algorithm in the sense that it can be applied with any feature vector accompanied with appropriate f_W. It is interesting to explore applications of the algorithm for ranking problems in other domains such as media (images, music) ranking.

References

1. Broder, A., Fontoura, M., Josifovski, V., Riedel, L.: A semantic approach to contextual advertising. In: SIGIR. ACM (2007)
2. Guha, R., McCool, R., Miller, E.: Semantic search. In: WWW. ACM (2003)
3. Green, S.: Building hypertext links by computing semantic similarity. IEEE 11 (1999)
4. Gabrilovich, E., Markovitch, S.: Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In: AAAI (2006)
5. Sun, X., Wang, H., Yu, Y.: Towards effective short text deep classification. In: SIGIR. ACM (2011)
6. Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Comp. Ling. 32, 13–47 (2006)
7. Bloehdorn, S., Moschitti, A.: Structure and semantics for expressive text kernels. In: CIKM. ACM (2007)
8. Gabrilovich, E., Markovitch, S.: Wikipedia-based semantic interpretation for natural language processing. AI Research (2009)
9. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: the concept revisited. In: WWW (2001)
10. Ponzetto, S.P., Strube, M.: Knowledge derived from wikipedia for computing semantic relatedness. JAIR (2007)
11. Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., Soroa, A.: A study on similarity and relatedness using distributional and wordnet-based approaches. In: NAACL (2009)
12. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: IJCAI (1995)
13. Jiang, J., Conrath, D.: Semantic similarity based on corpus statistics and lexical taxonomy. CoRR (1997)
14. Li, Y., Bandar, Z.A., McLean, D.: An approach for measuring semantic similarity between words using multiple information sources. IEEE 15, 871–882 (2003)
15. Jarmasz, M., Szpakowicz, S.: Roget's thesaurus and semantic similarity. In: RANLP (2003)
16. Tsatsaronis, G., Varlamis, I., Vazirgiannis, M., Nørvåg, K.: Omiotis: A Thesaurus-Based Measure of Text Relatedness. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 742–745. Springer, Heidelberg (2009)

17. Tsatsaronis, G., Varlamis, I., Vazirgiannis, M.: Text relatedness based on a word thesaurus. JAIR (2010)
18. Lesk, M.: Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In: SIGDOC, pp. 24–26. ACM (1986)
19. Banerjee, S., Pedersen, T.: Extended gloss overlaps as a measure of semantic relatedness. In: IJCAI (2003)
20. Patwardhan, S., Pedersen, T.: Using wordnet based context vectors to estimate the semantic relatedness of concepts. In: EACL (2006)
21. Morris, J., Hirst, G.: Lexical cohesion computed by thesaural relations as an indicator of the structure of text. Comp. Ling (1991)
22. Hirst, G., St-Onge, D.: Lexical chains as representations of context for the detection and correction of malapropisms. In: WordNet (1998)
23. Radinsky, K., Agichtein, E., Gabrilovich, E., Markovitch, S.: A word at a time: computing word relatedness using temporal semantic analysis. In: WWW (2011)
24. Milne, D., Witten, I.: An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In: AAAI Workshop (2008)
25. Yeh, E., Ramage, D., Manning, C., Agirre, E., Soroa, A.: Wikiwalk: random walks on wikipedia for semantic relatedness. ACL (2009)
26. Liberman, S., Markovitch, S.: Compact hierarchical explicit semantic representation. In: IJCAI Workshop (2009)
27. Haralambous, Y., Klyuev, V.: A Semantic Relatedness Measure Based on Combined Encyclopedic, Ontological and Collocational Knowledge. Computing Research Repository (2011)
28. Lin, D.: An information-theoretic definition of similarity. In: ICML (1998)
29. Dagan, I., Lee, L., Pereira, F.: Similarity-based models of cooccurrence probabilities. Machine Learning 34, 43–69 (1999)
30. Terra, E., Clarke, C.: Frequency estimates for statistical word similarity measures. In: NAACL (2003)
31. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. JASIST (1990)
32. Reisinger, J., Mooney, R.: Multi-prototype vector-space models of word meaning. In: NAACL, pp. 109–117 (2010)
33. Bollegala, D., Matsuo, Y., Ishizuka, M.: Measuring semantic similarity between words using web search engines. In: WWW (2007)
34. Eyke, H., Johannes, F., Weiwei, C., Klaus, B.: Label ranking by learning pairwise preferences. AI 172(16-17), 1897–1916 (2008)
35. Das Sarma, A., Das Sarma, A., Gollapudi, S., Panigrahy, R.: Ranking mechanisms in twitter-like forums. In: WSDM (2010)
36. Recchia, G., Jones, M.: More data trumps smarter algorithms: Comparing pointwise mutual information with latent semantic analysis. Behavior Research Methods (2009)
37. Cilibrasi, R., Vitanyi, P.: The google similarity distance. IEEE 19, 370–383 (2007)
38. Francis, W.N., Kucera, H.: Frequency analysis of English usage: Lexicon and grammer. Houghton Mifflin (1982)
39. Vapnik, V.: Statistical Learning Theory. Wiley Interscience (1998)
40. Rubenstein, H., Goodenough, J.B.: Contextual correlates of synonymy. Commun. ACM (1965)
41. Miller, G.A., Charles, W.G.: Contextual correlates of semantic similarity. Language and Cognitive Processes 6 (1991)
42. Hughes, T., Ramage, D.: Lexical semantic relatedness with random graph walks. In: EMNLP-CoNLL (2007)

Unsupervised Bayesian Part of Speech Inference with Particle Gibbs

Gregory Dubbin and Phil Blunsom

Department of Computer Science, University of Oxford, United Kingdom
Gregory.Dubbin@wolfson.ox.ac.uk,
Phil.Blunsom@cs.ox.ac.uk

Abstract. As linguistic models incorporate more subtle nuances of language and its structure, standard inference techniques can fall behind. These models are often tightly coupled such that they defy clever dynamic programming tricks. Here we demonstrate that Sequential Monte Carlo approaches, i.e. particle filters, are well suited to approximating such models. We implement two particle filters, which jointly sample either sentences or word types, and incorporate them into a Particle Gibbs sampler for Bayesian inference of syntactic part-of-speech categories. We analyze the behavior of the samplers and compare them to an exact block sentence sampler, a local sampler, and an existing heuristic word type sampler. We also explore the benefits of mixing Particle Gibbs and standard samplers.

1 Introduction

Modern research is steadily revealing more of the subtle structure of natural language to create increasingly intricate models. Recent advances in Bayesian non-parametrics have been employed by Computational Linguistics researchers to create effective unsupervised models of the latent structure text [1–4]. These models predominantly make use of the Dirichlet Process (DP), and its generalization the Pitman-Yor Process (PYP), in part due to the ease of deriving a collapsed Gibbs sampler for its inference. However this ease of inference comes at a cost; collapsed Gibbs samplers mix poorly due to the tight global dependencies present in the conditional distributions sampled from [5].

Previously [5] investigated techniques for alleviating the influence of long range conditional dependencies by simultaneously sampling groups of latent variables, however this approach is specific to models based on the simpler Dirichlet distribution. Here we present a general inference approach based on Sequential Monte Carlo (SMC) suitable for more advance models that employ Pitman-Yor Process priors.

Sequential Monte Carlo (SMC) methods, like particle filters, are particularly well suited to estimating tightly coupled distributions [6]. Particle filters sample sequences of latent variable assignments by concurrently generating several representative sequences consistent with a model's conditional probability distribution. The sequential nature of the sampling simplifies inference by ignoring ambiguous correlations with future latent variables at the cost of sampling the sequence multiple times. The few applications of particle filters in computational linguistics generally focus on the online nature of SMC [7, 8]. However, in this paper we demonstrate that batch applications benefit from the power of SMC to generate samples from tightly coupled distributions

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 760–773, 2012.

that would otherwise need to be approximated. Furthermore, the time cost of the additional samples generated by SMC can be mitigated by generating them in parallel.

In this paper we focus on the task of unsupervised part-of-speech (PoS) induction, where we seek to label tokens in a corpus with the syntactic role they play. In particular we describe and evaluate novel SMC inference algorithms for the Pitman-Yor Hidden Markov Model (PYP-HMM) first proposed in [4]. This model represents the state-of-the-art for unsupervised PoS induction, but the complex conditional dependencies present in the posterior led the authors of [4] to resort to a heuristic inference startegy and unrealistic restrictions on the model to achieve their highest reported results.

We start in Section 2 by introducing PYP-HMM model and its previously proposed inference algorithms. Section 3 introduces the Sequential Importance Sampling (SIS) algorithm, a basic SMC method that generates samples from the model's posterior. Using this approach we describe two novel inference algorithms for the PYP-HMM: a simple sentence-based block sampler (3.1) and a more complicated type-based sampler (3.2). We evaluate these algorithms in Section 4, analyzing their behavior in comparisons to the previously proposed state-of-the-art approaches. In summary we show that our SMC based algorithms are able to improve inference, finding higher probability modes, and task specific accuracies.

2 The Pitman-Yor Hidden Markov Model

The PYP-HMM model of PoS induction exhibits the tightly coupled correlations that complicate many standard inference methods [4]. The model applies a hierarchical Pitman-Yor process (PYP) prior to a trigram hidden Markov model (HMM) to jointly model the distribution of a sequence of latent word tags, $\mathbf{t}$, and word tokens, $\mathbf{w}$. The joint probability defined by the transition, $P_\theta(t_l|t_{n-1}, t_{n-2})$, and emission, $P_\theta(w_n|t_n)$, distributions of a trigram HMM is

$$P_\theta(\mathbf{t}, \mathbf{w}) = \prod_{n=1}^{N+1} P_\theta(t_l|t_{n-1}, t_{n-2}) P_\theta(w_n|t_n)$$

where $N = |\mathbf{t}| = |\mathbf{w}|$ and the special tag \$ is added to denote the sentence boundaries. The model defines the transition and emission distributions to be multinomial:

$$\begin{aligned} t_n|t_{n-1}, t_{n-2}, T &\sim T_{t_{n-1},t_{n-2}} \\ w_n|t_n, E &\sim E_{t_n} \end{aligned}$$

The PYP-HMM draws the above multinomial distributions from a hierarchical Pitman-Yor Process prior. The hierarchical prior can be intuitively understood to smooth the trigram transition distributions with bigram and unigram distributions in a similar manner to an ngram language model [9]. This backoff structure greatly reduces sparsity in the trigram distributions and is achieved by chaining together the PYPs through their base distributions:

$$
\begin{aligned}
T_{ij}|a^T, b^T, B_i &\sim \mathrm{PYP}(a^T, b^T, B_i) \\
B_i|a^B, b^B, U &\sim \mathrm{PYP}(a^B, b^b, U) \\
U|a^U, b^U &\sim \mathrm{PYP}(a^U, b^U, \mathrm{Uniform}). \\
E_i|a^E, b^E, C &\sim \mathrm{PYP}(a^E, b^E, C_i),
\end{aligned}
$$

where T_{ij}, B_i, and U are trigram, bigram, and unigram transition distributions respectively and C_i is either a uniform distribution (PYP-HMM) or a bigram character language model distribution (PYP-HMM+LM, intended to model basic morphology).

Draws from the posterior of the hierarchical PYP can be calculated with a variant of the Chinese Restaraunt Process (CRP) called the Chinese Restaurant Franchise (CRF) [9, 1]. In the CRP analogy, each latent variable (tag) in a sequence is represented by a customer entering a restaurant and sitting at one of an infinite number of tables. A customer chooses to sit at a table in a restaurant according to the probability

$$
P(z_n = k|\mathbf{z}_{1:n-1}) = \begin{cases} \frac{c_k^- - a}{n-1+b} & 1 \leq k \leq K^- \\ \frac{K^- a + b}{n-1+b} & k = K^- + 1 \end{cases} \tag{1}
$$

where z_n is the index of the table chosen by the nth customer to the restaurant, $\mathbf{z}_{1:n-1}$ is the seating arrangement of the previous $n-1$ customers to enter, c_k^- is the count of the customers at table k, and K^- is the total number of tables chosen by the previous $n-1$ customers. All customers at a table share the same dish, representing the value assigned to the latent variables. When customers sit at an empty table, a new dish is assigned to that table according to the base distribution of the PYP. To expand the CRP analogy to the CRF for hierarchical PYPs, when a customer sits at a new table, a new customer enters the restaurant representing the PYP of the base distribution.

Blunsom and Cohn [4] explored two Gibbs sampling methods for inference with the PYP-HMM model. The first individually samples tag assignments for each token. The second employs a tactic shown to be effective by earlier works by constraining inference to only one tag per word type (PYP-1HMM). However marginalizing over all possible table assignments for more than a single tag is intractable, and must be approximated. Blunsom and Cohn [4] approximates the PYP-1HMM tag assignment probabilities for a particular sample according to heuristic fractional table counts. Specific details of this model and associated samplers can be found in [4].

In this paper we present a principled SMC based sampler that allows for the simultaneous sampling of all the tags associated with a given word type without resorting to heuristics. This type based sampler achieves state-of-the-art performance without the requirement that all words of a given type share the same tag. This one-tag-per-type assumption is clearly false for syntactic categories (e.g. *sample* as a verb and a noun), thus its elimination is a step on the path to high performance unsupervised PoS induction.

3 Sequential Monte Carlo

Sequential Monte Carlo was introduced in 1993 as a Bayesian estimator for signal processing problems with strong non-linear conditional dependencies [10]. Since then,

SMC methods have been adopted by many fields, including statistics, biology, economics, etc. [11–13]. The SMC approach is the probabilistic analogue of the beam search heuristic, where the beam width can be compared to the number of particles and pruning is analogous to resampling. The basic SMC approach serves as the basis for several variants. Many SMC implementations resample the population of particles from the existing population to minimize the effect of increasing sample variance with increasing sequence length [14]. Particle smoothing variants of SMC reduce the relative variance of marginals early in the sequence, as well improving the diversity of the final sample [15]. Particle Markov chain Monte Carlo (PMCMC) formally augments classic Markov chain Monte Carlo (MCMC) approaches, like Gibbs sampling, with samples generated by particle filters [6].

While MCMC approximates a distribution as the average of a sequence of samples taken from the posterior of the distribution, SMC approximates a distribution as the importance weighted sum of several sequentially generated samples, called particles. This article describes two SMC samplers that jointly sample multiple tag assignments: a sentence based block sampler (`sent`) and a word type based block sampler (`type`). The basics of particle filtering are outlined below, while the implementation specifics of the `sent` and `type` particle filters are described in secions 3.1 and 3.2, respectively.

SMC is essentially the probabilistic analogue of the beam search heuristic. SMC stores P sequences, analogous to beam width, and extends each incrementally according to a proposal distribution q_n, similar to the heuristic cost function in beam search. Many particle filtering implementations also include a resampling step which acts like pruning by reducing the number of unlikely sequences.

We implemented Sequential Importance Sampling (SIS), detailed by Doucet and Johansen [16], to approximate joint samples from the sentence and word type distributions. This approach approximates a target distribution, $\pi_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n}$, of the sequence, $x_{1:n}$, of n random variables, that is $\gamma_n(x_{1:n})$ calculates the unnormalized density of $x_{1:n}$.

SIS initilizes each particle $p \in [1, P]$ by sampling from the initial proposal distribution $q_1(x_1^p)$, where x_n^p is the value assigned to the n-th latent variable for particle p. The algorithm then sequentially extends each particle according to the conditional proposal distribution $q_n(x_n^p|x_{1:n}^p)$, where $x_{1:n}^p$ is the sequence of values assigned to the first n latent variables in particle p. After extending a particle p, SIS updates the importance weight $\omega_n^p = \omega_{n-1}^p * \alpha_n(x_{1:n}^p)$. The weight update, defined as

$$\alpha_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1})q_n(x_n|x_{1:n-1})}, \tag{2}$$

accounts for the discrepancy between the proposal distribution, q_n, and the target distribution, π_n, without normalizing over $x_{1:n}$, which becomes intractable for longer sequences even in discrete domains. The normalizing constant of the target distribution is approximately $Z_n \approx \sum_{p=1}^{P} \omega_n^p$ and the unnormalized density is $\gamma_n(x_{1:n}) \approx \sum_{p=1}^{P} \omega_n^p \text{if} x_{1:n}^p = x_{1:n}$. The particles can also be used to generate an unbiased sample from π_n by choosing a particle p proportional to its weight ω_n^p.

Andrieu et al. [6] shows that to ensure the samples generated by SMC for a Gibbs sampler has the target distribution as the invariant density, the particle filter must be

modified to perform a *conditional SMC update*. This means that the particle filter gaurantees that one of the final particles is assigned the same values as the previous Gibbs iteration. Our implementation of the conditional SMC update reserves one special particle, 0, for which the proposal distribution always chooses the previous iteration's value at that site.

3.1 Sentence Sampling

The `sent` particle filter samples blocks of tag assignments $\mathbf{t}_{1:n}^{S}$ for a sentence, S, composed of tokens, $\mathbf{w}_{1:n}^{S}$. Sampling an entire sentence minimizes the risk of assigning a tag with a high probability given its local context but minimal probability given the entire sentence. Sentences can be sampled by ignoring table counts while sampling a proposal sentence, incorporating them after the fact with a Metropolis-Hastings acceptance test [17]. The Metropolis-Hastings step simplifies the sentence block particle filter further by not requiring the conditional SMC update.

While there is already a tractable dynamic programming approach to sampling an entire sentence based on the Forward-Backward algorithm, particle filtering the sentences PYP-HMM model should prove beneficial. For the trigram HMM defined by the model, the forward-backward sampling approach has time complexity in $O(NT^3)$ for a sentence of length N with T possible tag assignments at each site. Particle filters with P particles can approximate these samples in $O(NTP)$ time, which becomes much faster as T increases.

Sampling of sentence S begins by removing all of the transitions and emissions in S from the table counts, $\mathbf{z}$, resulting in the table counts $\mathbf{z}^{-S}$ of tag assignments $\mathbf{t}^{-S}$ the values assigned to the variables outside of S. For each site index $n \in [1, N]$ in the sentence, the particle filter chooses the new tag assignment, $t_n^{S,p}$, for each particle $p \in [1, P]$ from the sentence proposal distribution,

$$q_n^S(t_n^{S,p}|\mathbf{t}_{1:n-1}^{S,p}) \propto P(t_n^{S,p}|t_{n-2}^{S,p}, t_{n-1}^{S,p}, \mathbf{t}^{-S}, \mathbf{z}^{-S}) \times P(w_n^{S,p}|t_n^{S,p}, \mathbf{t}^{-S}, \mathbf{z}^{-S}, \mathbf{w}^{-S}).$$

After each new tag is assigned, the particle's weight is updated according to equation (2). The simplicity of the proposal density hints at the advantage of particle filtering over forward-backward sampling: it tracks only P histories and their weights rather than tracking the probability of over all possible histories. Once each particle has assigned a value to each site in the sentence, one tag sequence is chosen proportional to its particle weight, $\omega_N^{S,p}$.

3.2 Type Sampling

The type sampling case for the PYP-HMM is more complicated than the `sent` sampler. The long-range couplings defined by the hierarchical PYP priors strongly influence the joint distribution of tags assigned to tokens of the same word type [5]. Therefore, the affects of the seating decisions of new customers cannot be postponed during filtering as in sentence sampling. To account for this, the `type` particle filter samples sequences

of seating arrangements and tag assignments jointly, $\mathbf{x}^W_{1:n} = (\mathbf{t}^W_{1:n}, \mathbf{z}^W_{1:n})$, for the word-type, W. The final table counts are resampled once a tag assignment has been chosen from the particles.

Tracking the seating arrangement history for each particle adds an additional complication to the `type` particle filter. The exchangeability of seating decisions means that only counts of customers are necessary to represent the history. Each particle represents both a tag sequence, $\mathbf{t}^{W,p}_{1:n}$, and the count deltas, $\mathbf{z}^{W,p}_{1:n}$. The count deltas of each particle are stored in a hash table that maps a dish in one of the CRF restaurants to the number of tables serving that dish and the total number of customers seated at those tables. The count delta hash table ensures that it has sufficient data to calculate the correct probabilities (per equation (1)) by storing any counts that are different from the base counts, $\mathbf{z}^{-W}$, and defering to the base counts for any counts it does not have stored.

At each token occurence n, the next tag assignment, $t^{W,p}_n$ for each particle $p \in [1, P]$ is chosen first according to the word type proposal distribution

$$
\begin{aligned}
q^W_n(t^{W,p}_n | \mathbf{t}^{W,p}_{1:n-1}, \mathbf{z}^{W,p}_{1:n-1}) \propto \\
& P(t^{W,p}_n | c^{-2}_n, c^{-1}_n, \mathbf{t}^{-W,p}_{1:n-1}, \mathbf{z}^{-W,p}_{1:n-1}) \\
& \times P(c^{+1}_n | c^{-1}_n, t^{W,p}_n, \mathbf{t}^{-W,p}_{1:n-1}, \mathbf{z}^{-W,p}_{1:n-1}) \\
& \times P(c^{+2}_n | t^{W,p}_n, c^{+1}_n, \mathbf{t}^{-W,p}_{1:n-1}, \mathbf{z}^{-W,p}_{1:n-1}) \\
& \times P(w^W_n | t^{W,p}_n, \mathbf{t}^{-W,p}_{1:n-1}, \mathbf{z}^{-W,p}_{1:n-1}, \mathbf{w}^{-W,p}_{1:n-1}).
\end{aligned}
$$

In this case, $c^{\pm k}_n$ represents a tag in the context of site t^W_n offset by o, while $\mathbf{t}^{-W,p}_{1:n-1}$, $\mathbf{z}^{W,p}_{1:n-1}$, and $\mathbf{w}^{-W,p}_{1:n-1}$ represent the tag assignments, table counts, and word token values chosen by particle p as well as the values at all of the sites where a word token of type W does not appear. This proposal distribution ignores changes to the seating arrangement between the three transitions involving the site n. The specific seating arrangement of a particle is chosen after the tag choice, at which point the weights are updated by the result of equation (2). As with the `sent` sampler, once all of the particles have been sampled, one of them is sampled with probability proportional to its weight. This final sample is a sample from the true target probability.

As mentioned earlier, the sequence of particle approximations do not have the target distribution as invariant unless they use the conditional SMC update. Therefore, a special 0^{th} particle is automatically assigned the value from the prior iteration of the Gibbs sampler at each site n, though the proposal probability $q^W_n(t^{W,0}_n | \mathbf{t}^{W,p}_{1:n-1}, \mathbf{z}^{W,p}_{1:n-1})$ still has to be calculated to update the weight $\omega^{W,p}_n$ properly. This ensures that the `type` sampler has a non-zero probability of reverting to the prior iteration's sequence.

4 Experiments and Results

We aim to explore several aspects of both the PG sampler and the specifics of the PoS inference task. Firstly, the motivation of the PG implementations is to allow better inference on tightly coupled models. This suggests that the PG samplers should mix better than the local sampler, finding the mode of the model in fewer iterations. The type sampler should perform especially better with the character-LM, which assigns a higher

likelihood when words of the same type are labeled with the same tag. If the PG approach does find more likely modes, does that correspond to gains in practical measures like accuracy? Finally, how does the number of particles in PG samplers influence the inference? We hypothesize that increasing the number of particles decreases the variance of the posterior likelihoods as the chance of generating a population dominated by likely modes increases. However, the marginal improvement from additional particles should exhibit diminishing returns.

Section 4.1 describes the corpora on which these hypotheses are tested. We evaluate the performance of the samplers with two approaches. The first approach is an analysis of the samplers as inference algorithms. Each approach should tend toward a mode in the distribution as it mixes, resulting in more likely restaurant configurations. Section 4.2 analyzes the particle filter based samplers with various numbers of particles in an effort to understand how they behave. Then, section 4.3 evaluates each of the proposed approaches on PoS inference tasks from several languages. These results allow a practical comparison with other PoS inference approaches.

4.1 Data

Section 4.2 tests and compares several different approaches with a number of parameters. To simplify the procedures, all of the tests in the analysis are performed on a reduced version of the Penn Wallstreet Journal (WSJ) treebank, similar to Gao and Johnson [17] and Goldwater and Griffiths [18]. This reduced corpus is composed of the first 10,000 sentences in the Penn WSJ treebank, with 240,236 tokens. Additionally, this corpus uses the same, 17 tag, reduced tagset as Goldwater and Griffiths [18], developed by Smith and Eisner [19].

Section 4.3 compares the many-to-one (M-1) accuracy of the induced tag assignments on the full Penn WSJ treebank corpus, as well as the corpora from the CoNLL-X shared language task [20]. M-1 accuracy assigns the induced syntactic categories to the PoS of the most tokens of that category.

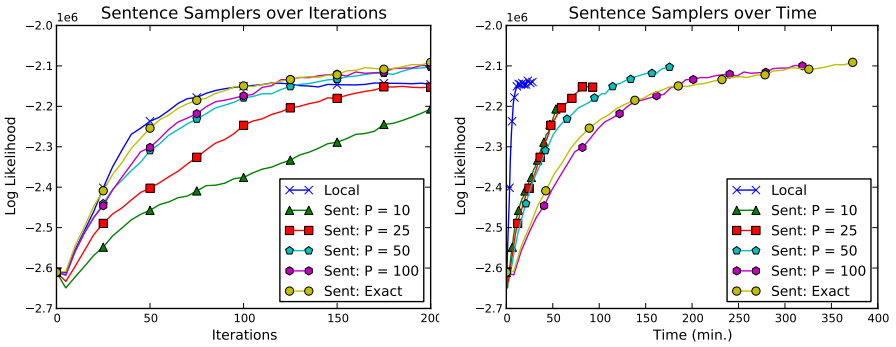
4.2 SMC Analysis

Before comparing the performance of the PG samplers to other inference methods, we wish to learn more about the approaches themselves. It is not obvious how well the benefits of block sampling transfer to SMC based approaches. Both the `sent` and `type` samplers are novel approaches to computational linguistics, and many of their properties are unclear. For example, the samples generated from the particle filter should have a higher variance than the target distribution. If the variance is too high, the sampler will be slower to converge. While additional particles lower the relative variance, they also increase the run time linearly. We hypothesize that there is a threshold of particles necessary to ensure that some are high likelihood sequences, beyond which inference gains are minimal the additional computational expense is wasted.

Like other MCMC methods, particle Gibbs generates a sequence of samples from the posterior distribution of the model in question. The PG sampler should mix more

quickly than a local sampler because it takes larger steps. These larger steps should move the PG sampler to a more likely mode in fewer iterations. The results in this section measure the power of the inference as the rate at which the posterior likelihood of the restaurant configuration increases.

The sentence based sampler, `sent`, samples from a distribution that can be exactly computed, facilitating comparisons between the exact sampler and the SMC approach. Figure 1 compares the posterior log-likelihoods of the `sent` sampler and the exact sentence sampler over 200 iterations. As expected, the likelihoods of the particle filters approach that of the exact sentence sampler as the number of particles increases from 10 to 100.

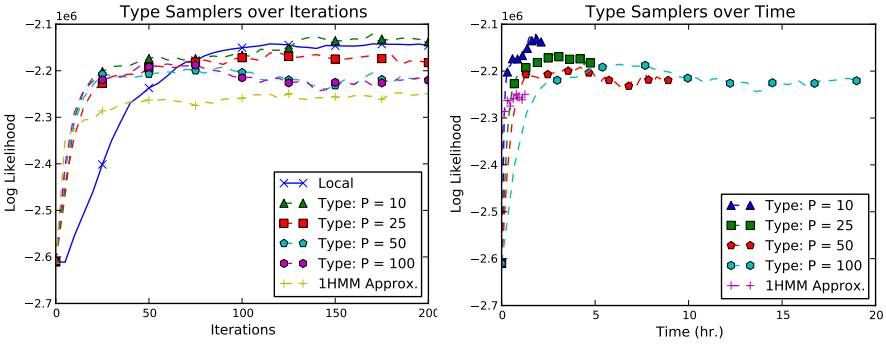


(a) Iterations (b) Time (min.)

Fig. 1. Posterior Log-Likelihood of PYP-HMM inference over iterations (a) and time (b) with exact as well as PG `sent` sampler with various numbers of particles. Except for the local sampler, which ran for 300 iterations in the time graph, the samplers were run for 200 iterations each, the end of a line represents the time to finish those 200 iterations.

Figure 2 compares the table configuration log-likelihood of the 1HMM approximation implemented by Blunsom and Cohn [4] with the `type` particle filter based sampler as well as the local sampler and the exact block sentence sampler. Unlike the sentence based block sampler, `type` sampler cannot be exactly calculated, even with the 1HMM approach of constraining inference to only consider sequences that assign the same tag to every token of the same word type. The 1HMM sampler approximates these probabilities using expected table counts. Theoretically, the `type` sampler should be a better approximation, being gauranteed to approach the true distribution as the number of particles increases. The results suggest that the `type` sampler can perform better than the 1HMM sampler with few particles. Unlike the `sent` sampler, the `type` sampler performs well even with fewer particles than the number of PoS categories in the tagset. These results suggest that the choice of block has a strong influence on the resulting sampler.

Interestingly, the local sampler reaches a higher mode than the 1HMM approach before the fiftieth iteration. However, earlier work by Blunsom and Cohn [4] found that the 1HMM approximation achieved a consistently higher M-1 accuracy than the local sampler. Section 4.3 confirms the same result. This reveals a disconnect between the likelihood under the PYP-HMM model and the M-1 accuracy. Additionally, both the `type` and 1HMM samplers reach likely modes within 30 iterations, after which they plateau and the additional cost of these approaches is wasted.



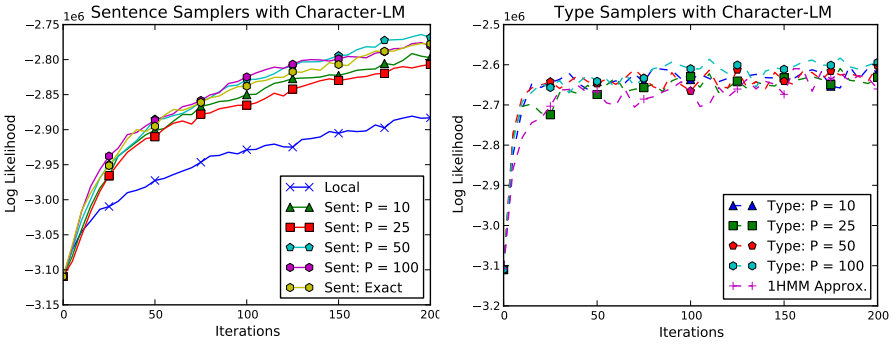
(a) Iterations (b) Time (hr.)

Fig. 2. Posterior Log-Likelihood of PYP-HMM inference over iterations (a) and time (b) with the `type` sampler as well as the 1HMM approximation proposed by Blunsom and Cohn [4] with various numbers of particles. Except for the local sampler, which ran for 300 iterations in the time graph, the samplers were run for 200 iterations each, the end of a line represents the time to finish those 200 iterations.

An interesting variant of the basic PYP-HMM model replaces the uniform base of the emission distribution with a bigram character language model (PYP-HMM-LM) [4]. In addition to allowing the PYP-HMM to recognize basic word morphology, the character language model creates a strong bias toward only one tag per word type. Figure 3(a) shows that the PYP-HMM-LM model is too tightly coupled for either the local or the `sent` samplers to mix quickly as they did with the simpler PYP-HMM model. The PYP-HMM-LM model reveals a weakness of the simple local Gibbs sampler approach, because the probability of a word being emitted by a new state is so low that sampler is highly likely to choose the same PoS tag assignment as other occurences of that word. The additional context of the `sent` sampler results in bigger steps and faster mixing than the local sampler, but most of the variables influencing probability of and particular tag assignment are in different sentences.

On the other hand, the `type` and 1HMM samplers sample all of the tags assignments for words of the same type simultaneously. Figure 3(b) shows that the result is drastically faster mixing. The `type` sampler is just as strongly influenced to assign all words of the same type to the same PoS category, but there are no same-word-type assignments

to bias the choice of which one tag they will all get. This strong correlation between the tags assigned to words of the same type may explain the `type` sampler's strong performance with so many fewer particles than necessary for the `sent` sampler: once a tag has been assigned to a sufficient number of words of the same type, that same tag will be progressively more likely to be chosen again. Perhaps a one-tag-per-type particle filter can take advantage of this fact to simplify the proposal distribution.



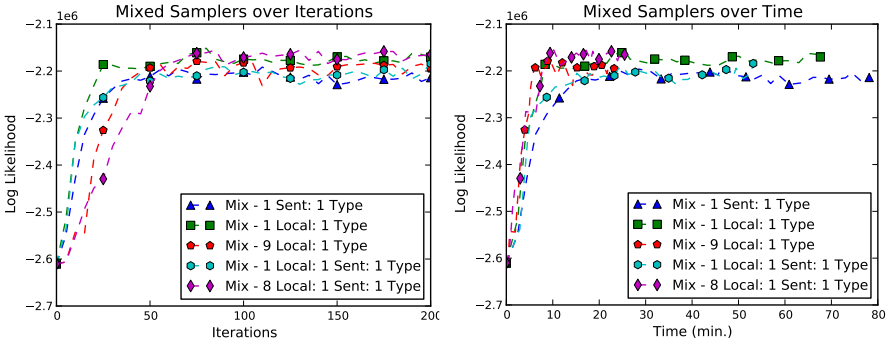
(a) `sent` (b) `type`

Fig. 3. Posterior Log-Likelihood of PYP-HMM-LM inference with the `sent` sampler (a), and the `type` sampler as well as the 1HMM approximation proposed by Blunsom and Cohn [4] (b) with various numbers of particles

The fact that the `type` sampler requires so few particles for inference relative to the `sent` sampler suggests that the choice of block can heavily influence the performance of a block Particle Gibbs sampler. The optimal block for a given model is not obvious and may be difficult to determine empirically. On the other hand, the large and expensive steps taken by a Particle Gibbs sampler are likely only necessary when faster methods reach modes from which they are slow to escape. A mixed sampler that only occasionally takes large steps with Particle Gibbs might achieve similar results with less computational cost. Such an approach need not even rely on any specific block partitioning, particle filters do not place many restrictions on the distributions from which they generate samples.

There are a multitude of possible combinations of mixed samplers, figure 4 demonstrates the performance of a few instantiations of one simple approach. Each of the mixed samplers in figure 4 randomly choses on of either the local, `sent`, or `type` samplers (the numbers in the legend describe the ratio of each sampler in the mixture.) For the sake of simplicity, each mixed sampler uses the same number of particles for the `sent` and `type` samplers, the mixed samplers in figure 4 all use ten particles. While there is a high degree of variance, the mixed samplers all show the rapid migration toward the mode shown by the `type sampler`, but each iteration takes less time on

average because of the sent and local samplers. The results in figure 4(b) suggest that the computational costs of the Particle Gibbs samplers can be mitigated by mixed sampling without eliminating the benefits from particle filtering. Finally, the samplers with the sent sampler in the mix performed quite well with ten particles despite the its poor performance with the same number of particles on its own. Perhaps mixing the block samplers diminishes the particle cost of poor block choice.



(a) Iterations (b) Time (min.)

Fig. 4. Posterior Log-Likelihood of PYP-HMM inference with various mixed samplers over 200 iterations ((a)) and over time ((b)). The colon separated lists provide the ratio of each mixed sampler: the sampler run on any given iteration is proportional to the number next to it. In each sampler, the particle filters are run using 10 particles. The samplers were run for 200 iterations each, the end of a line represents the time to finish those 200 iterations.

4.3 Unsupervised Part-of-Speech Tagging

Table 1 compares the M-1 accuracy of the sent and type particle filter samplers, from sections 3.1 and 3.2, with 100 particles each. Each site in a corpus is assigned the most commonly visited tag assignment at that site over all iterations. The particle filter based samplers rarely score a higher accuracy than even the local sampler, which completes 500 iterations before the particle filters complete 200.

While figures 1(a) and 2(a) show that all of the samplers surpass the 1HMM sampler in likelihood, the accuracies of the 1HMM and 1HMM-LM approximations remain well above the other approaches. This suggests that there are high-likelihood assignments that produce lower accuracy results, presumably related to the fact that the type sampler is not restricted to assignments with exactly one tag for each word type. If the model assigns equal likelihood to these assignments, inference will not be able to distinguish between them.

Excepting Arabic, the Type-10-LM sampler outperforms all of the other non-restrictive approaches, suggesting that the PYP-HMM-LM model may be a more accurate representation of PoS. As noted in section 4.2, the PYP-HMM-LM model is biased toward fewer tags per type than the more standard HMM model, resulting in an average number of tags per word type that is closer to the true value. Even so, figure 3(b) shows

Table 1. Many-to-1 accuracies on CoNLL and Penn-Treebank Wall Street Journal corpora for sentence- (Sent) and type- (Type) based filtering. The table lists the average M-1 accuracy measured according to the maximum marginal tag assignments over 3 seperate runs after 200 iterations for the `sent`, `type`, 1HMM and 1HMM-LM samplers, and 500 iterations for the HMM local sampler.

Language	Sent-100	Type-100	Type-10-LM	Local	1HMM	1HMM-LM	Tokens	Tag types
WSJ	69.8%	70.1%	74.7%	70.2%	75.6%	77.5%	1,173,766	45
Arabic	53.5%	57.6%	51.3%	56.2%	61.9%	62.0%	54,379	20
Bulgarian	64.8%	67.8%	71.6%	67.6%	71.4%	76.2%	190,217	54
Czech	59.8%	61.6%	65.2%	64.5%	65.4%	67.9%	1,249,408	12^c
Danish	65.0%	70.3%	74.9%	69.1%	70.6%	74.6%	94,386	25
Dutch	61.6%	71.6%	70.1%	64.1%	73.2%	72.9%	195,069	13^c
Hungarian	61.8%	61.8%	68.5%	64.8%	69.6%	73.2%	131,799	43
Portuguese	59.4%	71.1%	74.4%	68.1%	72.0%	77.1%	206,678	22
Spanish	66.3%	69.1%	75.3%	68.5%	74.7%	78.8%	89,334	47
Swedish	62.9%	63.5%	68.3%	67.6%	67.2%	68.6%	191,467	41

that the `type` sampler mixes at least as well as the 1HMM sampler with the character language model, yet the 1HMM sampler still scored a higher many-to-one accuracy on all languages other than Danish. This discrepancy between model likelihood and accuracy suggests that a different model is necessary to further improve unsupervised PoS induction.

5 Conclusion

This paper presented a novel application of the Particle Gibbs sampler approach to the computational linguistic inference application of unsupervised PoS induction. Such approaches show great potential for inference, especially in highly dependent distributions e.g. non-parametric Bayesian applications. While this power generally comes and the expense of significantly increased computation, results show that a mixed sampler that only occasionally performs a Particle Gibbs sampling step can achieve similar results in a fraction of the time. Additionally, the type particle filter itself can be largely run in parallel, only bottlenecking when the particle weights need to be normalized. Further expansion of the basic ideas presented will enable scalable inference in otherwise intractable models.

References

1. Goldwater, S., Griffiths, T., Johnson, M.: Interpolating between types and tokens by estimating power-law generators. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems 18, pp. 459–466. MIT Press, Cambridge (2006)
2. Liang, P., Petrov, S., Jordan, M., Klein, D.: The infinite PCFG using hierarchical Dirichlet processes. In: Proc. of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP 2007), Prague, Czech Republic, pp. 688–697 (2007)

3. Cohn, T., Goldwater, S., Blunsom, P.: Inducing compact but accurate tree-substitution grammars. In: HLT-NAACL 2009: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 548–556. Association for Computational Linguistics, Morristown (2009)
4. Blunsom, P., Cohn, T.: A hierarchical Pitman-Yor process hmm for unsupervised part of speech induction. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 865–874. Association for Computational Linguistics, Portland (2011)
5. Liang, P., Jordan, M.I., Klein, D.: Type-based MCMC. In: North American Association for Computational Linguistics, NAACL (2010)
6. Andrieu, C., Doucet, A., Holenstein, R.: Particle markov chain monte carlo methods. Journal of The Royal Statistical Society Series B 72(3), 269–342 (2010)
7. Canini, K.R., Shi, L., Griffiths, T.L.: Online inference of topics with latent Dirichlet allocation. In: van Dyk, D., Welling, M. (eds.) Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS 2009), pp. 65–72 (2009)
8. Borschinger, B., Johnson, M.: A particle filter algorithm for bayesian wordsegmentation. In: Proceedings of the Australasian Language Technology Association Workshop 2011, Canberra, Australia, pp. 10–18 (December 2011)
9. Teh, Y.W.: A hierarchical bayesian language model based on Pitman-Yor processes. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics. ACL-44, pp. 985–992. Association for Computational Linguistics, Morristown (2006)
10. Gordon, N.J., Salmond, D.J., Smith, A.F.M.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings F Radar and Signal Processing 140(2), 107–113 (1993)
11. Jasra, A., Doucet, A., Stephens, D., Holmes, C.: Interacting sequential Monte Carlo samplers for trans-dimensional simulation. Computational Statistics & Data Analysis 52(4), 1765–1791 (2008)
12. Beaumont, M.A.: Estimation of Population Growth or Decline in Genetically Monitored Populations. Genetics 164(3), 1139–1160 (2003)
13. Fernandez-Villaverde, J., Rubio-Ramirez, J.F.: Estimating macroeconomic models: A likelihood approach. Review of Economic Studies 74(4), 1059–1087 (2007)
14. Kitagawa, G.: Monte carlo filter and smoother for non-gaussian nonlinear state space models. Journal of Computational And Graphical Statistics 5(1), 1–25 (1996)
15. Fearnhead, P., Wyncoll, D., Tawn, J.: A sequential smoothing algorithm with linear computational cost. Technical report, Department of Mathematics and Statistics, Lancaster University (2008)
16. Doucet, A., Johansen, A.M.: A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later. Oxford University Press (2009)
17. Gao, J., Johnson, M.: A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, pp. 344–352. Association for Computational Linguistics, Morristown (2008)
18. Goldwater, S., Griffiths, T.: A fully bayesian approach to unsupervised part-of-speech tagging. In: Proc. of the 45th Annual Meeting of the ACL (ACL 2007), Prague, Czech Republic, pp. 744–751 (June 2007)

19. Smith, N.A., Eisner, J.: Contrastive estimation: Training log-linear models on unlabeled data. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL), Ann Arbor, Michigan, pp. 354–362 (June 2005)
20. Buchholz, S., Marsi, E.: Conll-x shared task on multilingual dependency parsing. In: Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL-X 2006, pp. 149–164. Association for Computational Linguistics, Morristown (2006)

WikiSent: Weakly Supervised Sentiment Analysis through Extractive Summarization with Wikipedia

Subhabrata Mukherjee and Pushpak Bhattacharyya

Dept. of Computer Science and Engineering, IIT Bombay
{subhabratam,pb}@cse.iitb.ac.in

Abstract. This paper describes a *weakly* supervised system for sentiment analysis in the movie review domain. The objective is to classify a movie review into a polarity class, *positive* or *negative*, based on those sentences bearing opinion on the movie alone, leaving out other irrelevant text. *Wikipedia* incorporates the world knowledge of *movie-specific features* in the system which is used to obtain an *extractive summary* of the review, consisting of the reviewer's opinions about the specific aspects of the movie. This filters out the concepts which are irrelevant or objective with respect to the given movie. The proposed system, *WikiSent*, does not require any labeled data for training. It achieves a better or comparable accuracy to the existing semi-supervised and unsupervised systems in the domain, on the same dataset. We also perform a general movie review *trend analysis* using WikiSent.

Keywords: Sentiment Analysis, Wikipedia, Information Extraction, Weakly Supervised System, Text mining, Summarization, Reviews.

1 Introduction

In the movie domain, like many other product domains, there has been a flurry of review sites giving critics view about the performance of the actor, director, story as well as the public acceptance of the movie. This is of importance not only to the people directly related to the movie-making but also to the audience, whose viewing decisions are quite influenced by these reviews.

Sentiment analysis of movie reviews aims to automatically infer the opinion of the movie reviewer and often generates a rating on a pre-defined scale. Automated analysis of movie reviews is quite a challenge in text classification due to the various nuances associated with the critic reviews. The author may talk about a lot of topics which are not directly related to the movie in focus. Tightly intermixed with various objective statements are his subjective opinions about the movie, which are quite difficult to extract. Here, an objective statement is defined as not just a factual statement, but as objective from the point of view of analyzing the opinion about a particular movie.

This work is different from traditional automatic text summarization or abstractive summarization. This is because the objective is not to obtain a shorter text but to retrieve *relevant opinionated text*. This focused extraction requires external world knowledge about the various technical aspects of the movie (like *movie plot, film*

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 774–793, 2012.

crew, characters, domain specific features etc.). Wikipedia feeds the system with this technical knowledge which is used to create an extract of the review. This extract is subsequently classified by a lexicon. Figure 1 shows the system architecture.

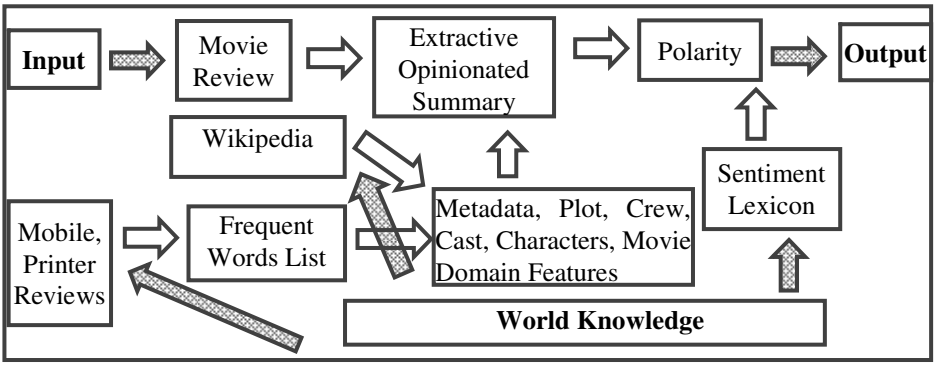


Fig. 1. System Block Diagram

Consider the fragment of a review of the movie L.I.E taken from the IMDB movie review corpus [14] which has been tagged as a negative review:

Example 1. Review of the Movie L.I.E

[1]Best remembered for his understated performance as Dr. Hannibal Lecter in Michael Mann's forensics thriller, Manhunter, Scottish character actor Brian Cox brings something special to every movie he works on. [2]Usually playing a bit role in some studio schlock (he dies halfway through The Long Kiss Goodnight), he's only occasionally given something meaty and substantial to do. [3]If you want to see some brilliant acting, check out his work as a dogged police inspector opposite Frances McDormand in Ken Loach's Hidden Agenda.

[4]Cox plays the role of Big John Harrigan in the disturbing new indie flick L.I.E., which Lot 47 picked up at Sundance when other distributors were scared to budge. [5]Big John feels the love that dares not speak its name, but he expresses it through seeking out adolescents and bringing them back to his pad. [6]What bothered some audience members was the presentation of Big John in an oddly empathetic light. [7]He's an even-tempered, funny, robust old man who actually listens to the kids' problems (as opposed to their parents and friends, both caught up in the high-wire act of their own confused lives.). [8]He'll have sex-for-pay with them only after an elaborate courtship, charming them with temptations from the grown-up world"

……

[9]It's typical of unimaginative cinema to wrap things up with a bullet, sparing the writers from actually having to come up with a complex, philosophical note. [10]In this regard, l.i.e. (and countless other indie films) share something in common with blockbuster action films: problems are solved when the obstacle is removed. [11]How often does real life work this way to extend the question : if a movie is striving for realism , do dramatic contrivances destroy the illusion?

The first paragraph of the review talks about the central character Brian Cox's notable performance in some earlier movie. The second paragraph gives a brief description of his character in an empathetic light which comprises of positive opinions about the character. The reviewer opinion about the movie comes only in the last paragraph, where he gives some negative opinions about the movie. The review consists of majority positive words, not all of which are significant to the reviewer opinion, outweighing the negative opinions about the movie. A bag-of-words classifier, thus, would wrongly classify this as positive.

In this work, we give an analysis of the various aspects of a movie review in Section 3. There we highlight the significant and non-significant concepts for sentiment analysis of movie reviews. Section 4 describes how the sectional information in Wikipedia helps in this task. It also gives the automated feature extraction process from Wikipedia. Section 5 gives the algorithm for the extraction of the opinion summary consisting of the *relevant* reviewer statements, which is classified by a sentiment lexicon in Section 6. Section 7 discusses different approaches for parameter learning for the model. The experimental evaluation is presented in Section 8 on a gold standard dataset of 2000 movie reviews as well as on an unlabeled pool of 27,000 documents to find the trend. Section 9 discusses the results followed by conclusions in Section 10.

The main contribution of this work is to show how Wikipedia info-box sectional information can be used to incorporate *World Knowledge* in a system, to obtain an *extractive opinionated summary* of a movie review. This, in turn, helps in sentiment analysis of the review due to the filtering out of objective concepts from subjective opinions. This work is *mostly* unsupervised, requiring no labeled training data. The weak supervision comes from the usage of resources like WordNet, POS-tagger and Sentiment Lexicons, due to their mode of construction.

2 Related Work

There are 2 prominent paradigms in automatic text summarization [3]: *extractive* and *abstractive* text summarization. While *extractive* text summarization attempts to identify prominent sections of a text by giving more emphasis on the content of the summary, *abstractive* text summarization gives more emphasis on the form so that the sentences are syntactically and semantically coherent. The *topic-driven* summarization paradigm is more common to IR where the summary content is based on the user query about a particular topic. [4] attempts to find the top-ranked significant sentences based on the frequency of the content words present in it. [5] gives importance to the position of a sentence i.e. where the sentence appears in the text and comes up with an optimum position policy and emphasis on the cue words. [6] uses tf-idf to retrieve signature words, NER to retrieve tokens, shallow discourse analysis for cohesion and also uses synonym and morphological variants of lexical terms using WordNet. [7] uses a rich set of features like *Title, Tf & Tf-Idf scores, Position score, Query Signature, IR Signature, Sentence Length, Average Lexical Connectivity, Numerical Data, Proper Name, Pronoun & Adjective, Weekday & Month, Quotation, First Sentence* etc. and uses decision trees to learn the feature weights. There are other works based on HMM [8], RTS [9], lexical chain and cohesion [10].

We use many of the above features for finding the extract of the summary. However, our objective differs in the fact that we intend to derive relevant *subjective sentences* significant for the movie, and not objective sentences. It is also topic-driven, depending on the *movie plot, actors, film crew, fictional characters* etc.

[11] proposes to find subjective sentences using lexical resources where the authors hypothesize that subjective sentences will be more similar to opinion sentences than to factual sentences. As a measure of similarity between two sentences they used different measures including shared words, phrases and the WordNet. [12] focuses on extracting top sentiment keywords which is based on Pointwise Mutual Information (PMI) measure [13].

The pioneering work for subjectivity detection is done in [14], where the authors use min-cut to leverage the *coherency* between the sentences. The fundamental assumption is that local proximity preserves the objectivity or subjectivity relation in the review. But the work is completely supervised requiring two levels of tagging. Firstly, there is tagging at the sentence level to train the classifier about the subjectivity or objectivity of individual sentences. Secondly, there is tagging at the document level to train another classifier to distinguish between positive and negative reviews. Hence, this requires a lot of manual effort. [15] integrates graph-cut with linguistic knowledge in the form of WordNet to exploit similarity in the set of documents to be classified.

Now, if a system possesses world knowledge about the technical aspects of the movie, then it would be easier for it to detect objective or subjective sentences based on the key concepts or features of a movie. Wikipedia[1] can incorporate this world knowledge in the system. Wikipedia is recently used in a number of works mainly for concept expansion in IR for expanding the query signature [16], [17], [18] as well as for topic driven multi document summarization [19].

There has been a few works in sentiment analysis using Wikipedia [20], [21]. [20] focuses on concept expansion using Wikipedia where they expand the feature vector constructed from a movie review with related concepts from the Wikipedia. This increases accuracy as it helps in unknown concept classification due to expansion, but it *does not* address the concern of separating subjective from objective sentences.

These works do not take advantage of the *Sectional* arrangement of the Wikipedia articles into categories. Each Wikipedia movie article has sections like *Plot*, *Cast, Production etc.* which can be explicitly used to train a system about the different aspects of a movie. In this work, our objective is to develop a system that requires no labeled data for training and classifies the *opinionated extractive summary* of the movie; where the summary is created based on the extracted information from Wikipedia.

3 Facets of a Movie Review

Movie review analysis is a challenging domain in Sentiment Analysis due to sarcasm, thwarting and requirement of extensive world knowledge. The reviewer opinion about the movie may target the characters in the movie, the plot or his expectations from the crew involved. We categorize the reviewer statements in the following categories:

[1] http://www.wikipedia.org/

Table 1. Reviewer Statement Categories

1. General Perception about the Crew	6. Opinion about Movie Characters
2. Objective Facts about the Crew and Movies	7. Characteristics of a Movie or Genre
3. Past Performance of the Crew and Movies	8. Opinion about the Movie and Crew
4. Expectations from the Movie or Crew	9. Unrelated Category
5. Movie Plot	

We define *Crew* in a movie as the people who are involved in making of the movie like the *Producer, Director, Actor, Story-Writer, Cinematographer, Musician* etc. We are mainly interested in extracting opinions from *Category 8* where all the other Categories may lend a supporting role to back his opinions or add noise. We give examples (taken from the movie reviews of the IMDB corpus [2]) for each of the categories in *Table 2*.

Table 2. Reviewer Statement Categories with Examples

1.	General Perception about the Crew	*John Travolta is considered by many to be a has-been, or a one-hit wonder … Leornardo DeCaprio is an awesome actor.*
2.	Objective Facts about the Crew and Movie	*Born into a family of thespians -- parents Roger Winslet and Sally Bridges-Winslet were both stage actors – Kate Winslet came into her talent at an early age.*
3.	Past Performance of the Crew	*The role that transformed Winslet from art house attraction to international star was Rose DeWitt Bukater, the passionate, rosy-cheeked aristocrat in James Cameron's Titanic (1997).*
4.	Expectations from the Movie or Crew	*I cancelled the date with my girlfriend just to watch my favorite star featuring in this movie.*
5.	Movie Plot	*L.I.E. stands for Long Island Expressway, which slices through the strip malls and middle-class homes of suburbia. Filmmaker Michael Cuesta uses it as a metaphor of dangerous escape for his 15-year old protagonist, Howie (Paul Franklin Dano).*
6.	Opinion about the Characters in the Movie	*He's an even-tempered, funny, robust old man who actually listens to the kids' problems (as opposed to their parents and friends, both caught up in high-wire act of their confused lives.).*
7.	Characteristics of a Movie or Genre	*Horror movies are supposed to be scary.* *There is an axiom that directors who have a big hit with their debut have a big bomb with their second film.*
8.	Opinion about the Movie and Crew	*While the movie is brutal, the violence is neither very graphic nor gratuitous. It may scare the little ones, but not any teen-ager. Besides the awesome direction, the ageless glamor and fabulous acting of Leonardo DeCaprio and Kate Winslet made the movie titanic a timeless hit.*
9.	Unrelated Category	*So my grandson gives me passes to this new picture One Night at McCool's because the free screening is the same night as that horrible show with those poor prisoners trapped on the island who eat the bugs. "Go," he says, "it's just like Rush-o-Man."*

It is evident from the examples above why movie domain text is difficult to analyze. Consider the Example from *Category 5*, which talks about the movie plot. The keyword dangerous, there, makes the segment negative. But it expresses only a concept about the movie and not the reviewer opinion. Similarly, *Category 6 Example* talks about a character in the movie which expresses a positive opinion but unrelated w.r.t the opinion analysis of the review. *Category 7 Example* has the keywords *movie* and *audience* directly related to the movie domain. Thus it is more probable that they are expressing some direct opinion about a certain aspect of the movie. Similarly, the name of the *actors* in *Category 8 Example 2* makes it relevant, as they reflect opinions about the persons involved in the making of the movie. Hence, it is important to extract only those concepts which are significant from the point of view of opinion analysis of the movie and filter out the non-significant portion. A unigram based bag-of-words model would capture a lot of noise, if it considers all categories to be equally relevant.

4 Wikipedia Information Extraction for Movie Review Analysis

Wikipedia is the largest English knowledge repository consisting of more than 20 million articles collaboratively written by volunteers all around the world. It is open-access and regularly updated by about 100,000 active contributors daily. Each Wikipedia page is an article on some known concept or topic. Each article belongs to one of the many defined categories or subcategories. For Example, Category Film has 31 sub-categories like *Film making, Works about Films, Film Culture* etc. Furthermore, each article has a number of sections which can be edited separately. Any Wikipedia article on films may consist of sections like *Plot, Cast, Production, Marketing, Release* etc. which are common among most of the articles of that category. We utilize this feature to extract movie specific information from the Wikipedia.

A Wikipedia movie article consists of a small paragraph, in the beginning, giving information about the movie story, crew and achievements in short. We call this section the *Metadata* about the movie. There is a table on the extreme right of the article which provides information about the name of the *Producer*, *Director, Actors, Cinematographer* etc. We call this section the *Crew* Information. There is a section on the movie plot which summarizes the story of the movie and talks about its fictional aspects. We call this section the *Plot* of the movie. There is another section which gives information about the actors in the movie, the roles they perform and the characters they enact. We call this section the *Character* of the movie. We use all the above information extracted from the Wikipedia article about the particular movie to incorporate World Knowledge into the system.

We used the IMDB movie corpus [2] consisting of 2,000 tagged positive and negative movie reviews, each class consisting of 1,000 reviews. The tagged information is used only for evaluation. Furthermore, there is a collection of 27,000 raw html

documents taken from the IMDB review site, from which the authors extracted and tagged the above 2000 documents, used for finding the general trend in the movie domain.

4.1 Wikipedia Article Retrieval

The 2,000 processed review documents had their titles removed. Thus the corresponding reviews had to be retrieved from the unprocessed html documents and their titles extracted. The title of the movie review was used to construct a *http get* request to retrieve the corresponding html page of the movie directly from Wikipedia. In case of multiple articles in different domains with same name, the *film* tag was used to retrieve the desired article. For multiple movies with the same name, the year (in which the movie was released) information available with the title was used to extract the correct Wikipedia article. Thus the Wikipedia article retrieval was in the order *film name* → *film tag* → *film year*.

4.2 Crew Information

All the crew information were extracted from the table in the right hand side of the Wiki article, bearing the name of all the persons involved in the making of the movie like the *director, producer, cinematographer, story-writer etc.*, and added to the **Crew** list.

The first line in the Wiki article that contains the phrase *Directed by* or *Author* and is a part of any table (detected by the html tags */td, /tr, /th*, */table etc.*) is taken as the start of the Crew info-section. The phrases *Release Date* or *Language* or the html tags */table* and */tbody*, that signify the end of the Crew table, is taken as the end of the info-section.

4.3 Metadata Extraction

The metadata was extracted from the html page just below the title of the article. The text was POS-tagged using a part of speech tagger[2] and all the *Nouns* were extracted. The Nouns were further stemmed[3] and added to the **Metadata** list. The words were stemmed so that *acting* and *action* have the same entry corresponding to *act*. Some movie articles in Wikipedia had the Plot section missing. The metadata was used in those cases to replace the Plot. In other cases, the *Metadata* information was simply appended to the *Plot* information.

According to the structure of the Wikipedia html page on movie articles, the metadata information on the movie appears just after the Crew table in the html page.

[2] http://nlp.stanford.edu/software/tagger.shtml

[3] http://sourceforge.net/projects/stemmers/files/Lovins-stemmer-Java/

This section spans the page from the end of the Crew info-section till the start of the next info-box, which is indicated by the Wiki tag *editsection.* The Wiki tag *editsection* allows users to edit an info-box. All the Wikipedia info-boxes are editable.

4.4 Plot Extraction

The movie plot was extracted from the *Plot Section* of the Wikipedia. The words were extracted similarly as in Metadata extraction. In both the Plot and Metadata, the concepts were restricted to be *Nouns*. For example, in the Harry Potter movie the nouns *wizard, witch, magic, wand, death-eater, power* etc. depict concepts in the movie.

The Wikipedia html id-attribute *id="Plot"* is taken as the beginning of the Plot info-box which spans the text till the next info-box, indicated by the Wiki tag *editsection.*

If we consider all the Nouns, a lot of noise will be incorporated into the **Plot** list. This is because the commonly used Nouns like *vision, flight, inform, way* etc. are added as well. To prevent this, both in the *Metadata* and the *Plot*, a separate list was created comprising of the frequently found terms (it will be shortly discussed how this list was compiled) in a corpus. Subsequently, the frequently occurring words were filtered out, leaving only *movie* and *genre-specific* concepts in the Metadata and Plot list.

4.5 Character Extraction

The **Cast** section in the Wiki article has the name of the actors and the name of the characters they enact. These character names were extracted and added to the **Character** list. These depict the fictional roles played by the actors in the movie.

The Wiki html id-attribute *id="Cast"* is taken as the beginning of the Cast info-box which spans the text till the next info-box, indicated by the Wiki tag *editsection.*

4.6 Frequent Word List Construction

The underlying hypothesis for this list creation is that movie reviews will have certain *concepts and terms* those are exclusive to this domain and will less frequently occur in other domains. Review data from the *Printer* and *Mobile Phone* domains[4] were used to create a list of frequently occurring terms in those domains. Since those domains are completely disjoint from the movie review domain, words which *frequently* occur in all of these domains must be commonly occurring words. Thus the commonly used words list consists of the frequently occurring terms in all these domains. The *tf-idf* measure was used and all those words above a threshold were added to the **FreqWords** list. For example, the word *person* (which is a Noun) occurred in all the domains with a very high frequency and thus added to the **FreqWords** list.

[4] `http://mllab.csa.iisc.ernet.in/downloads/reviewmining/fulldata.tar.gz`

4.7 Domain Specific Feature List Construction

Wikipedia articles on films and aspects of films[5] were extracted. The sentences in those documents were POS-tagged. The Nouns were retrieved and frequently occurring words were removed. The remaining words were stemmed and added to the **MovieFeature** list. *Table 3* shows a snapshot of the genre specific terms extracted from Wiki movie articles.

Table 3. Extracted Movie Domain Specific Terms

Movie, Staffing, Casting, Writing, Theory, Rewriting, Screenplay, Format, Treatments, Scriptments, Synopsis, Logline, Pitching, Certification, Scripts, Budget, Ideas, Funding, Plans, Grants, Pitching, Tax, Contracts, Law, Copyright, Pre-production, Budgeting, Scheduling, Preproduction, Film, Stock, Story, Boarding, Plot, Directors, Location, Scouting, …..

5 Algorithm to Extract Opinion Summary

Section 4 describes the creation of the feature lists *Metadata, Plot, Crew, Character, FreqWords* and *MovieFeature*. Now, given a movie review the objective is to extract all the sentences that reflect the true opinion of the reviewer about the movie. This forms the *OpinionSummary* of the movie. A sentence-by-sentence analysis of a review is performed.

Any sentence not involving any word from any of the above lists is not considered relevant at all, thus pertaining to the *Unrelated Category 9*. Sentences involving concepts from the *Plot, Metadata* and *Character Lists* are considered least significant, as they talk about the movie story and not about the reviewer opinion. But they are not considered completely irrelevant as they may contain sentences that back the reviewer's opinion about the movie. This covers *Category 5 and 6*. Sentences containing terms from the *MovieFeature* list are likely to comment on some specific aspect of the movie and are thus considered more relevant (*Category 7*). Finally, any sentence containing the movie *Title*, or *Crew* information is considered most relevant, as the reviewer is likely to express his opinion about the movie. This covers *Category 1-4*. *The final opinion of the reviewer (Category 8) is actually a weighted function of all the other Categories.*

The *Metadata, Plot and Character* lists are combined into a single list called the *Plot*. We now have 3 main categories of features corresponding to the *Plot, Crew* and *MovieFeature* lists with an auxiliary *FreqWords* list.

Given a movie review R with n sentences S_i, our objective is to determine whether each sentence S_i is to be *accepted* or *rejected* based on its *relevance* to the reviewer opinion. Let each sentence S_i consist of n_i words $w_{ij}, j \in 1 \ldots n_i$. The *Plot* list does not contain any word from the *FreqWords* list or the *MovieFeature* list. Similarly, the *MovieFeature* list also does not contain any word from the *FreqWords* list. The *relevance factor* of the sentence S_i is given by,

[5] http://en.wikibooks.org/wiki/Movie_making_manual

$$Rel_{factor_i} = \alpha \sum_j 1_{w_{ij} \in Crew \text{ or } w_{ij} \in MovieTitle} + \beta \sum_j 1_{w_{ij} \in MovieFeature} - \gamma \sum_j 1_{w_{ij} \in Plot, w_{ij} \notin Crew, w_{ij} \notin MovieTitle}$$
$$where \ \alpha, \beta, \gamma > 0, \qquad \alpha > \beta \tag{1}$$

The relevance factor is actually a weighted combination of the various *features* in the *lists*. It *counts* the words appearing from different lists in a sentence and *weighs* them separately. The concepts belonging to the *Plot* are not so much relevant in judging the reviewer's opinion about the movie and may add noise. They play a dampening role, which is captured in the '-' sign before γ. More weight is given to any word referring to the *Crew* or *Movie Title* than any word simply referring to a *movie domain feature*, which is captured in $\alpha > \beta$. Any sentence S_i is accepted if Acc_{factor_i} corresponding to S_i is 1.

$$Acc_{factor_i} = 1 \ \textbf{if} \ Rel_{factor_i} \geq \theta \ and \ \exists w_{ij} \in S_i \ \boldsymbol{s.t.} \ w_{ij} \in Crew \ or \ MovieFeature \ or \ Movie \ Title$$
$$= 0 \ otherwise \tag{2}$$

Here θ is a threshold parameter. Thus any sentence is accepted as being relevant, if its score is greater than some threshold value and there is *atleast one* word in the sentence that belongs to the *Crew*, *MovieFeature* or the *MovieTitle* lists.

Considering the Review *Example 1*, the algorithm works as follows: Let us consider α, β, γ to assume integer values. Let $\alpha = 2, \beta = 1, \gamma = 1$. The variables assume the first integer values satisfying all the conditions in *Equation 1*. *Let* $\theta = 0$.

In Sentence [1], *Brian Cox* is the only keyword present and it belongs to the *Cast* list (the other keywords are not present in the Wiki article for the film L.I.E.).

Rel_{factor_1}=2*1+1*0-1*0=2 $\geq$ 0, $Acc_{factor_1} = 1$ and the sentence is accepted. In [2], there is no keyword from the lists and it is rejected straightaway. [3] has the keyword *acting* from MovieFeature and is accepted where, Rel_{factor_3}=1, $Acc_{factor_1} = 1$. [4] has the keywords *Cox, L.I.E* from *Cast and MovieTitle*, *John Harrigan* from *Character* list and *distributor* from the *MovieFeature* list. Rel_{factor_4}=2*2+1-1=4 $\geq$ 0 and the sentence is accepted. [5] has only the keyword *Big John* from *Character.* Its Rel_{factor_5}=0+0-1=-1$\not\geq$ 0 and the sentence is rejected. [6] has the keyword *audience* from *MovieFeature* and *Big John* from *Character*. Its Rel_{factor_6}=0+1-1=0 $\geq$ 0 and it is accepted. [7] has the keywords *temper, friend* from *Plot* and Rel_{factor_7}=0+0-2=-2 $\not\geq$ 0 and is rejected. [8] has the keywords *sex, charm* from *Plot* and Rel_{factor_8}=0+0-2=-2 $\not\geq$ 0 and is rejected. [9] has the keywords *cinema, writers* from *MovieFeature* and *bullet* from *Plot*. Thus Rel_{factor_9}=0+1*2-1=1 $\geq$ 0 and is accepted as being relevant. [10] has the keywords *l.i.e* from *MovieTitle, films(2), action* from *MovieFeature*. Thus $Rel_{factor_{10}}$=2*1+1*3-0=5 $\geq$ 0 and is accepted as being relevant. [11] has the keyword *movie* from *MovieFeature*. $Rel_{factor_{11}}$=0+1*1-0=1 $\geq$ 0 and it is accepted.

Algorithm 1. Extractive Opinion Summary from Review

Input : Review R
Output: OpinionSummary
Step 1: Extract the Crew list from Wikipedia
Step 2: Extract the Plot list from Wikipedia
Step 3: Extract the MovieFeature list from Wikipedia
Step 4: Extract the FreqWords list as the common frequently occurring concepts in Mobile Phone, Printer and Movie domains.
Let $OpinionSummary = \emptyset$
for i=1..n
  if $Acc_{factor_i} == 1$
    add S_i *to* $OpinionSummary$

6 Classification of the Opinion Summary

The words in the extracted opinion summary can be directly used as features in a supervised classification system. But since we do not use any labeled data for training, a sentiment lexicon is used in the final phase to classify the opinion summary. A sentiment lexicon contains an opinion word along with its polarity. SentiWordNet [22], Inquirer [23] and the Bing Liu [24] sentiment lexicons are used to find the polarity of a word. SentiWordnet is tagged at the synset level (*word sense and polarity*) whereas the Inquirer and Bing Liu sentiment lexicons contain the words and their most commonly considered polarity. While using the SentiWordNet, we use the first sense of a word as we do not perform word sense disambiguation.

Let $pol(w_{ij})$ be the polarity of a word w_{ij}, where i indexes a particular sentence and j indexes a particular word in the sentence. Let $flip_{ij}$ be a variable which indicates whether the polarity of w_{ij} should be flipped or not. Negation handling is being done in the lexical classification, in which the polarity of all the words in a window of 5, from the occurrence of any of the negation operators *not, neither, nor and no*, are flipped.

The final polarity of the review (*pos* or *neg*) is given by,

$$sign\left(\sum_{i=1}^{m}\sum_{j=1}^{n_i} flip_{ij} \times pol(w_{ij}) \times g\left(sign\left(flip_{ij} \times pol\left(w_{ij}\right)\right)\right)\right)$$
$$where\ g(x) = negation\_bias\ if\ x < 0$$
$$= 1 \qquad if\ x \geq 0 \tag{3}$$

The polarity is given by the signed sum of the polarity bearing opinion words in the sentence weighted by the *negation_bias*. The range of the polarity function is $[-m\ \ m]$, where m is the number of polarity-bearing words in the sentence.

Any review has more explicit positive expressions of opinion than negative ones [1],[25],[26],[27]. This is because negative sentiment is often implicit as in sarcasm and thwarting. Likewise sentiment lexicons have a high bias towards positive and

objective words [1]. A negation bias is added, so that the occurrence of any negative word in the review is weighed more than a positive word. The SentiWordNet has a high coverage as well as a high bias towards positive and objective terms since it uses a semi-supervised learning method. Inquirer, being manually hand-tagged, has a low coverage but high accuracy similar to the Bing Liu sentiment lexicon, although the latter has a higher coverage than the Inquirer. We experimented with all the *three* lexicons.

7 Parameter Setting

A simple but effective strategy used in information retrieval and automatic text summarization for feature weighting is to use weights that are simple integral multiples, preferably prime, to reduce the possibility of ties [35]. There are 5 parameters for the model we used: $\alpha, \beta, \gamma, \theta$ *and* $negation_{bias}$. The first 3 parameters can be best trained if sentence level label (whether each sentence is relevant or not) information is available. However, in the absence of any label information, we adopt a simpler approach as mentioned above. We took the first set of integer values, satisfying all the constraints in *Equation 1*, and assigned them to the first 3 parameters : $\boldsymbol{\alpha = 2}$, $\boldsymbol{\beta = 1}, \boldsymbol{\gamma = 1}$.

The value of θ should be set such that the number of significant keywords, from *Crew, MovieFeature and MovieTitle* lists, should be more than the number of keywords from less significant concepts like *Plot* and *Character* lists. This means Rel_{factor_i} should be greater than or equal to zero which, implies $\theta = 0$.

The authors in [1] weighted up the negative expressions by a fixed amount (50%) over positive expressions. In our experiment, value of the *negation bias* is determined as:

$$negation\_bias = \frac{positive\ opinion\ words\ in\ the\ corpus}{negative\ opinion\ words\ in\ the\ corpus}$$

This is done to give any positive or negative opinion word equal importance in the review. In the ideal case, since the corpus is balanced having equal number of positive and negative reviews, the ratio should have been close to 1, in absence of any negation bias. However, due to the bias problem explained before, the *negation_bias* comes out to be 1.4.

Semi-supervised Learning of Parameters

This work *does not* evaluate this angle for parameter learning, since our objective has been to develop a system that requires no labeling information at the sentence level or the document level. However, if some sentence level information is available, a robust learning of parameters is possible.

Equation 1 and 2 can be re-written as:

$$Rel_{factor_i} = \alpha \times X_{i,1} + \beta \times X_{i,2} - \gamma \times X_{i,3}$$
$$Acc_{factor_i} = Rel_{factor_i} - \theta$$
$$= \alpha \times X_{i,1} + \beta \times X_{i,2} - \gamma \times X_{i,3} - \theta$$
$$= \alpha \times X_{i,1} + \beta \times X_{i,2} - \gamma \times X_{i,3} - \theta \times X_{i,4} \text{ (where } X_{i,4} = 1)$$

Let Y_i be the binary label information corresponding to each sentence in the development set, where Y_i=*1* if $Acc_{factor_i} \geq 0$ and -1 otherwise.

$Y_i = \boldsymbol{W^T}.X_i^T$ where,

$\boldsymbol{W} = [\alpha\ \beta\ -\gamma\ -\theta]^T$ *and* $X_i = [X_{i,1}\ X_{i,2}\ X_{i,3}\ X_{i,4}]$

$or, \boldsymbol{Y} = \boldsymbol{W^T}.\boldsymbol{X}$ *where*, $\boldsymbol{X} = [X_1^T\ X_2^T \ldots X_n^T]$ *and* $\boldsymbol{Y} = [Y_1\ Y_2 \ldots Y_n]$

This is a linear regression problem which can be solved by the ordinary least squares method by minimizing the sum of the squared residuals i.e. the sum of the squares of the difference between the observed and the predicted values [37]. The solution for $\boldsymbol{W}$ is given by $W = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T Y}$ **.**

A regularizer can be added to protect against over-fitting and the solution can be modified as:

$\boldsymbol{W} = (\boldsymbol{X^T X} + \delta \boldsymbol{I})^{-1}\boldsymbol{X^T Y}$ *where* δ *is a parameter and* $\boldsymbol{I}$ *is the identity matrix.*

8 Evaluation

The experimental evaluation is performed on the IMBD movie review corpus [2]. It consisted of 2000 reviews collected from the IMDB movie review site and polarity labeled at the document level, 1000 from each of the two classes. This forms our gold standard data. Apart from this, there is an unlabeled corpus of 27,886 unprocessed html files from which the above 2000 reviews had been extracted and labeled by the annotators.

The parameters are set as: $\boldsymbol{\alpha = 2, \beta = 1, \gamma = 1, \theta = 0, negation\_bias = 1.4}$.

8.1 Movie Review Analysis Using WikiSent

This analysis is performed on the unprocessed pool of 27,886 html documents. The movie reviews belong to 20 different genres. *Figure 2* shows the number of movies belonging to each genre in the dataset as well as all the genre names.

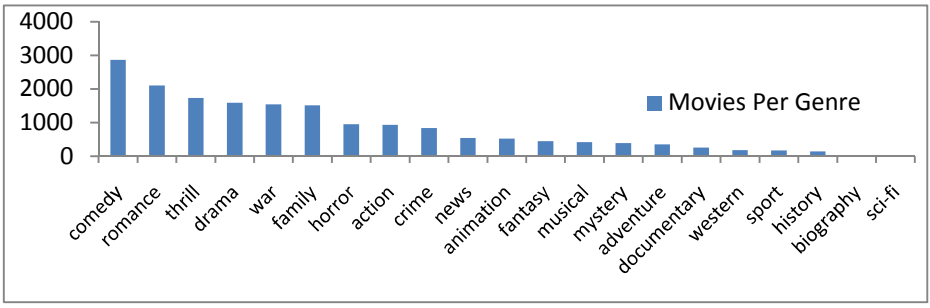


Fig. 2. Movies per Genre in the Dataset

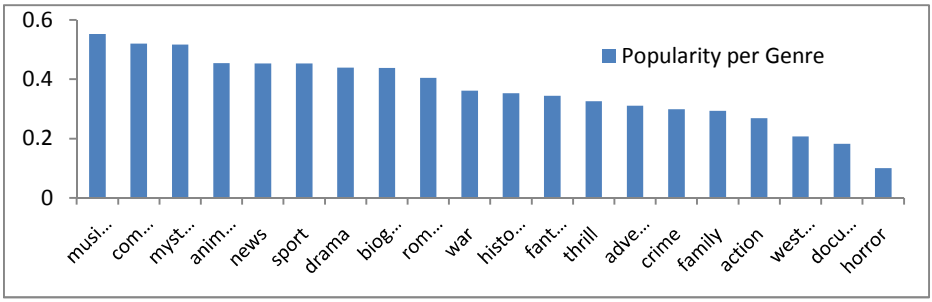


Fig. 3. Genre Popularity in the Dataset

The genre popularity (refer to *Figure 3*) is given by:

$$Genre\ Popularity = \frac{Positive\ Movie\ Reviews\ per\ Genre}{Total\ Movie\ Reviews\ per\ Genre}$$

Table 4 gives the fraction of the movie reviews that are predicted to be positive and negative by WikiSent and the baseline bag-of-words model (expressed in *percentage*). *Figure 4* shows the total number of movies released and the total number of movies predicted to be positive *per year* (as is present in the dataset).

Table 4. Movie Review Polarity Comparison of WikiSent vs. Baseline System

	WikiSent	Bag-of-Words Baseline
Positive Reviews (%)	48.95	81.2
Negative Reviews (%)	51.05	18.79

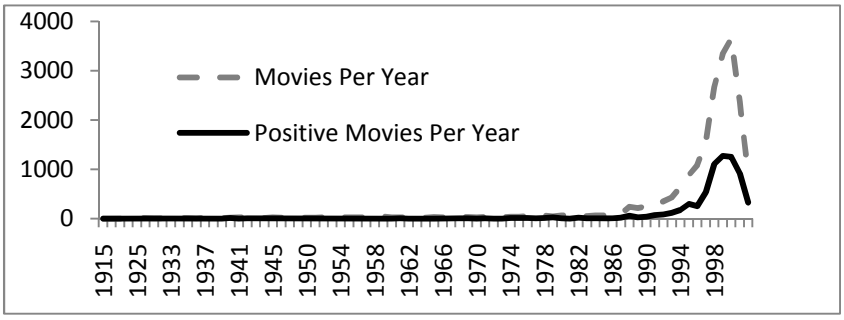


Fig. 4. Movie Popularity per Year in the Dataset

8.2 WikiSent Evaluation on the Gold Standard Data

The baseline for WikiSent has been taken as the bag-of-words model, in which all the terms in a review are considered relevant and classified with the help of a lexicon. The baseline accuracy *Table 5* is adapted from [1], in which the evaluation is done on the same dataset as ours, but with different sentiment lexicons. It also shows the performance of *WikiSent* using different sentiment lexicons. *Table 6* shows the accuracy variation with $\boldsymbol{\theta}$ and *Table 7* shows the accuracy variation with *negation_bias*. *Table 8* shows that accuracy comparison of WikiSent with the *best performing unsupervised and semi-supervised systems* in the domain. The compared systems have been evaluated on the *same* corpus [2]. We directly incorporated the accuracies from the respective papers for comparison.

Table 5. Accuracy using Different Lexicons Without and With WikiSent

Only Google-Full	66.31	*Only* Subjectivity-Full	65.42
Only Google-Basic	67.42	*Only* Subjectivity-Basic	68.63
Only Maryland-Full-Now	67.42	*WikiSent* + SentiWordNet	73.30
Only Maryland-Basic	62.26	**WikiSent + Inquirer (GI)**	**76.85**
Only GI-Full	64.21	*WikiSent* + Bing Liu	69.80
Only GI-Basic	65.68	*WikiSent* + Above 3 Lexicons	74.56
Only SentiWordNet-Full	61.89		
Only SentiWordNet-Basic	62.89		

Table 6. Accuracy Variation with $\boldsymbol{\theta}$

$\boldsymbol{\theta}$	Accuracy
3	63.63
2	66.74
1	69.31
0	**76.85**
-1	71.43

Table 7. Accuracy Variation with ***negation_bias***

Negation_Bias	Accuracy
1	70.89
1.1	73.90
1.3	74.74
1.4	**76.85**
1.5	75.53
1.6	73.59

Table 8. Accuracy Comparison with Different Systems

System	Classification Method	Accuracy
Li [33]	Semi Supervised with 10% doc. label	60.00
Li [33]	Semi Supervised with 40% doc. label	60.00
Lin [32] LSM	Unsupervised without prior info	61.70
Taboada SO-CAL Basic [1]	Lexicon Generation	68.05
Shi [28], Dasgupta [29]	Eigen Vector Clustering	70.90
Lin [32] LSM	Unsupervised with prior info	74.10
Taboada SO-CAL Full [1]	Lexicon Generation	76.37
Socher [30] RAE	Semi Supervised Recursive Auto Encoders with random word initialization	76.80
WikiSent	**Wikipedia+Lexicon**	**76.85**
Nakagawa [31]	**Supervised** Tree-CRF	77.30
Socher [30] RAE	Semi Supervised Recursive Auto Encoders with 10% cross-validation	77.70

9 Discussions

9.1 Movie Trend Analysis

The movie review corpus contains most movies from the genres *comedy, romance, thrill, drama, war, horror, action, crime* and least number of movies from the genres *west, sport, history, biography, sci-fi* (in the descending order). This depicts a general trend in the movie-making of sticking to the most popular genres.

It is observed that movies belonging to the categories *musical, comedy, mystery, animation* and *news* received the most number of positive reviews whereas movies belonging to the genres *family, action, western, documentary* and *horror* received the least number of positive reviews. This shows that there are a large number of movies from the *comedy* genre and in general these movies tend to do well; whereas the movies from the *action and horror* genres, despite having a large number of releases, do not fare very well. The movies from the genres *musical and animation*, generally have a good acceptance despite less number of releases.

The number of movies per year has grown exponentially with time as is observed from *Figure 4*. This also highlights the importance of movie review analysis in the socio-economic aspect. It is seen that the number of movies as well as good movies have increased with time. The dip after the year 2000 may be attributed to the fact that the data collection process was only till 2002, so the reviews crawled after 2000 were less.

The number of negative reviews in the movie domain actually outweighs the number of positive reviews, to some extent (refer to *Table 4*). This shows that people are a bit skeptical in calling a movie *good*, despite the large number of movies that are

being released. It is also seen that the baseline bag-of-words model, which tags a *huge* number of movie reviews as positive, is unable to catch this trend. This also shows the limitation of the baseline model which considers all words to be relevant, in analyzing movie reviews.

9.2 WikiSent Performance Analysis

It is observed that *WikiSent* performs the best with *Inquirer* (GI) among all the other lexicons used with it. It is interesting to find the huge accuracy leap from the baseline accuracy using *Only SentiWordNet* (**61.89 and 62.89**) and *SentiWordNet + WikiSent* (**73.3**) in *Table 5*. This accuracy improvement is achieved through the deployment of extractive summarization using Wikipedia, by which the objective sentences are eliminated from the review before classification. However, using all the resources (3) together does not give the maximum accuracy.

As the value of $\boldsymbol{\theta}$ increases, fewer sentences are considered relevant due to which many informative sentences are left out. Higher value of $\boldsymbol{\theta}$ means a single sentence should have a large number of representatives from the *Crew*, *MovieFeature* lists, which is rare. Again, as $\boldsymbol{\theta}$ decreases more number of sentences are considered relevant which captures noise due to the inclusion of many insignificant sentences. Low value of $\boldsymbol{\theta}$ means the number of insignificant features from the *Character, Plot* lists outnumber those from the *Crew*, *MovieFeature* lists.

As *negation_bias* increases, negative expressions outweigh positive expressions and accuracy decreases. A low value of the *negation_bias* is unable to offset the inherent corpus bias of the positive expressions and accuracy falters.

WikiSent achieves a better accuracy than most of the existing unsupervised and semi-supervised systems, as is evident from *Table 8*. Its performance is comparable to *SO-Cal Full* [1], *Recursive Auto Encoders* (RAE) [30] and *Tree-CRF* [31]. The accuracy difference of WikiSent with these systems is *not* statistically signficant. The SO-Calculator does not use any document label like WikiSent, whereas the Tree-CRF is supervised and RAE [30] reports a 10-fold cross-validation. It is notable that WikiSent is able to perform better or at par with the semi-supervised systems, which use partial document labels, without using any labeled training data.

9.3 WikiSent Drawbacks

One of the biggest drawbacks of the system is that we do not perform co-reference resolution due to which valuable information is lost. Thus any sentence having a feature anaphorically referring to a relevant feature in the previous sentence will be ignored, due to which significant sentences may be rejected. We do not perform word-sense disambiguation[6], in this work. Since we consider only the first sense of the word (which is not always the best sense according to the context) we miss out on the actual sense of a word and its proper polarity in many cases [36]. For example, we use a simple lexicon which does not distinguish between the various meanings of the

[6] `http://en.wikipedia.org/wiki/Word-sense_disambiguation`

same word, like '*bank*' in the sense of '*relying'* which is a *positive term* and '*bank*' in the sense of a '*river bank*' which is *objective*. Furthermore the lexicon, that we use, has a low coverage. If a more specialized lexicon had been used, like SO-CAL [1], more accuracy improvement would have been possible. *Inquirer* suffers from a low coverage since it is manually hand-tagged. Thus many of the polarity-bearing words are absent in it. Though SentiWordNet has a large coverage, it is biased towards positive and objective terms and classifies less number of words as negative.

10 Conclusions and Future Work

In this work, we proposed a weakly supervised approach to sentiment classification of movie reviews. The polarity of the review was determined by filtering out irrelevant objective text from relevant subjective opinions *about the movie in focus*. The relevant opinionated text extraction was done using Wikipedia.

We showed that the incorporation of world knowledge through Wikipedia, to filter out irrelevant objective text, can significantly improve accuracy in sentiment classification. Our approach differs from other existing approaches to sentiment classification using Wikipedia in the fact that *WikiSent* does not require any labeled data for training. Weak supervision comes from the usage of resources like WordNet, POS-Tagger and Sentiment Lexicons. This work is different in the way it creates an *extractive opinionated summary* of the movie using the *sectional* information from Wikipedia and then uses a lexicon to find its polarity. The work extensively analyzes the significance of the various aspect specific features in movie reviews that are relevant to sentiment analysis and harnesses this information using Wikipedia. We define an *acceptance factor* for each sentence in the review based on which it should be included in the *extract* or not. In the final stage, we use a simple sentiment lexicon to classify the words in the extract to find the final polarity (*positive or negative*) of the review.

WikiSent has a number of parameters which have been simplistically set in the absence of any label information. In case the polarity information is used, the parameters can be set robustly (the semi-supervised learning method describes this aspect) which may further increase accuracy. The system suffers from the lack of handling *anaphora resolution* and *word sense disambiguation*. Usage of a simple lexicon at the final stage for polarity calculation also mars its accuracy. Addressing these concerns, significant performance improvement may be possible.

Nevertheless, we showed that *WikiSent* attains a *better* or *comparable accuracy* to all the existing unsupervised and semi-supervised systems in the domain on the same dataset, without using any labeled data for training. Furthermore, we also do a general analysis of the movie domain using WikiSent (based on the *genre, year of release* and *movie review polarity*) to show the general trends persisting in movie-making as well as public acceptance of the movie.

References

1. Taboada, M., Brooke, J., Tofiloski, M., Voll, K., Stede, M.: Lexicon-based methods for sentiment analysis. In: Computational Linguistics 2011 (2011)
2. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment Classification using Machine Learning Techniques. In: Proceedings of EMNLP 2002 (2002)
3. Das, D., Martins, A.F.T.: A Survey on Automatic Text Summarization, Literature Survey for the Language and Statistics II course at CMU, Pittsburg (2007)
4. Luhn, H.P.: The automatic creation of literature abstracts. IBM Journal of Research Development 2(2), 159–165 (1958)
5. Edmundson, H.P.: New methods in automatic extracting. Journal of the ACM 16(2), 264–285 (1969)
6. Aone, C., Okurowski, M.E., Gorlinsky, J., Larsen, B.: A trainable summarizer with knowledge acquired from robust nlp techniques. In: Mani, I., Maybury, M.T. (eds.) Advances in Automatic Text Summarization, pp. 71–80 (1999)
7. Lin, C.-Y.: Training a selection function for extraction. In: Proceedings of CIKM 1999, pp. 55–62 (1999)
8. Conroy, J.M., O'leary, D.P.: Text summarization via hidden markov models. In: Proceedings of SIGIR 2001, pp. 406–407 (2001)
9. Marcu, D.: Improving summarization through rhetorical parsing tuning. In: Proceedings of The Sixth Workshop on Very Large Corpora, Montreal, Canada, pp. 206–215 (1998)
10. Barzilay, R., Elhadad, M.: Using lexical chains for text summarization. In: Proceedings ISTS 1997 (1997)
11. Yu, H., Vasileios, H.: Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In: EMNLP (2003)
12. Potthast, M., Becker, S.: Opinion Summarization of Web Comments. In: Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., van Rijsbergen, K. (eds.) ECIR 2010. LNCS, vol. 5993, pp. 668–669. Springer, Heidelberg (2010)
13. Turney, P.: Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002 (2002)
14. Pang, B., Lee, L.: A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In: Proceedings of the ACL (2004)
15. Agarwal, A., Bhattacharyya, P.: Sentiment Analysis: A New Approach for Effective Use of Linguistic Knowledge and Exploiting Similarities in a Set of Documents to be Classified. In: International Conference on Natural Language Processing (ICON 2005), IIT Kanpur, India (December 2005)
16. Müller, C., Gurevych, I.: Using Wikipedia and Wiktionary in Domain-Specific Information Retrieval. In: Peters, C., Deselaers, T., Ferro, N., Gonzalo, J., Jones, G.J.F., Kurimo, M., Mandl, T., Peñas, A., Petras, V. (eds.) CLEF 2008. LNCS, vol. 5706, pp. 219–226. Springer, Heidelberg (2009)
17. Wu, F., Weld, D.S.: Automatically Refining the Wikipedia Infobox Ontology. In: WWW (2008)
18. Milne, D.N., Witten, I.H., Nichols, D.M.: A knowledge-based search engine powered by wikipedia. In: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management. ACM, New York (2007)
19. Wang, H., Zhou, G.: Topic-driven Multi-Document Summarization. In: Proceedings International Conference on Asian Language Processing (2010)

20. Gabrilovich, E., Markovitch, S.: Overcoming the Brittleness Bottleneck using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. In: Proceedings of the 21st National Conference on Artificial Intelligence, vol. 2. AAAI (2006)
21. Wang, P., Domeniconi, C.: Building semantic kernels for text classification using Wikipedia. In: KDD (2008)
22. Baccianella, S., Esuli, A., Sebastiani, F.: Sentiwordnet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In: LREC 2010 (2010)
23. Stone, P.J., Dunphy, D.C., Smith, M.S., Ogilvie, D.M., Associates: The General Inquirer: A Computer Approach to Content Analysis. The MIT Press (1966)
24. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Seattle, Washington, USA, August 22-25 (2004)
25. Kennedy, A., Inkpen, D.: Sentiment classification of movie and product reviews using contextual valence shifters. Computational Intelligence 22(2), 110–125 (2006)
26. Voll, K., Taboada, M.: Not all words are created equal: Extracting semantic orientation as a function of adjective relevance. In: Proceedings of the 20th Australian Joint Conference on Artificial Intelligence, Gold Coast, pp. 337–346
27. Boucher, J.D., Osgood, C.E.: The Pollyanna hypothesis. Journal of Verbal Learning and Verbal Behaviour 8, 1–8 (1969)
28. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
29. Dasgupta, S., Ng, V.: Topic-wise, Sentiment-wise, or Otherwise? Identifying the Hidden Dimension for Unsupervised Text Classification. In: EMNLP (2009)
30. Socher, R., Pennington, J., Huang, E.H., Ng, A.Y., Manning, C.D.: Semi-supervised recursive autoencoders for predicting sentiment distribution. In: EMNLP (2011)
31. Nakagawa, T., Inui, K., Kurohashi, S.: Dependency tree-based sentiment classification using CRFs with hidden variables. In: NAACL (2010)
32. Lin, C., He, Y., Everson, R.: A comparative study of Bayesian models for unsupervised sentiment detection. In: CoNLL (2010)
33. Li, T., Zhang, Y., Sindhwani, V.: A nonnegative matrix tri-factorization approach to sentiment classification with lexical prior knowledge. In: Proceedings of (ACL-IJCNLP), pp. 244–252 (2009)
34. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
35. McCarthy, D., Koeling, R., Weeds, J., Carroll, J.: Finding Predominant Word Senses in Untagged Text. In: Proceedings of the 42nd Meeting of the Association for Computational Linguistics, ACL 2004 (2004)
36. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc. (2006)

Adaptive Two-View Online Learning for Math Topic Classification

Tam T. Nguyen, Kuiyu Chang, and Siu Cheung Hui

Nanyang Technological University,
50 Nanyang Avenue, Singapore 639798
nguy0080@e.ntu.edu.sg, {askychang,asschui}@ntu.edu.sg

Abstract. Text categorization has been a popular research topic for years and has become more or less a practical technology. However, there exists little research on math topic classification. Math documents contain both textual data and math expressions. The text and math can be considered as two related but different views of a math document. The goal of online math topic classification is to automatically categorize a math document containing both mathematical expressions and textual content into an appropriate topic without the need for periodically retraining the classifier. To achieve this, it is essential to have a two-view online classification algorithm, which deals with the textual data view and the math expression view at the same time. In this paper, we propose a novel adaptive two-view online math document classifier based on the Passive Aggressive (PA) algorithm. The proposed approach is evaluated on real world math questions and answers from the Math Overflow question answering system. Compared to the baseline PA algorithm, our method's overall F-measure is improved by up to 3%. The improvement of our algorithm over the plain math expression view is almost 6%.

1 Introduction

Math documents such as math questions, scientific papers, etc., constitute a substantial portion of modern scientific literature. To organize these materials for easy retrieval, they are usually classified into pre-defined categories via automatic topic classifiers. Online documents such as blog posts, question and answer posts, emails, etc., are growing at extreme speeds. Manual classification is time consuming and expensive, to say the least. Over the years, machine-based topic classification has become matured enough to be deployed practically for this task. However, existing methods only focus on the textual data, which typically overwhelms underlying semantics like math expressions, tables, charts, diagrams, etc. In an attempt to improve the performance of classifiers on rich math content documents, we propose a novel approach for math-aware topic classification.

Clearly, it is trivial to regard math expressions as normal text data during tokenization, and treat the entire math document just like a regular text document using conventional text document classification methods. However, this approach basically ignores math expressions, which are highly structured data

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 794–809, 2012.

containing valuable hints. As such, math expression semantics should be extracted using math-aware methods. Ideally, we should treat text and math as two distinct feature sets or views. To classify math documents based on text and math features, we need a suitable algorithm that can work on the two kinds of data at the same time without one view dominating the other. SVM-2K [11] and Two-view SVM [17] can be applied in this case. However, in dynamic systems such as online question answering sites, where new data is generated continuously, an online/incremental classification algorithm is more desirable. Such a system can predict the topic of each posting and adjust dynamically (if the automatic prediction is deemed wrong by the user). In this case, an online learning algorithm such as Perceptron [2,21], Second-order Perceptron [4], or Passive Aggressive (PA) [8] can be considered. However, these algorithms only work on a single view and therefore cannot be applied directly to two-view data.

User postings in question answering systems such as *Cross Validated*[1], *Meta Optimize*[2], and *Math Overflow*[3], etc., typically contain many math expressions. While Cross Validated and Meta Optimize are mainly used by computer scientists, Math Overflow users include serious mathematicians. Moreover, one common characteristic of postings on all three sites is their rich math content. To automatically classify these content, we need a fast online learning algorithm that can work on two kinds of features, textual data and math expressions.

2 Related Work

Document topic classification aims to automatically categorize a given document into the appropriate topics or classes. Common classification algorithms include Naïve Bayes [15,16], Nearest-Neighbors [6], C4.5 [22], Support Vector Machine (SVM) [5], etc. Document topic classification has been applied to many domains, e.g., emails, blogs, and news articles.

For online document topic classification, many algorithms have been proposed. The Perceptron algorithm [2,21] is simple and fast but its classification accuracy is not good enough. To improve the performance of the Perceptron algorithm, Cesa-Bianchi et al. [4] proposed the Second-order Perceptron (SOP) algorithm, which takes advantage of second-order information; it performs better than the original Perceptron in terms of accuracy but is slower due to the added complexity required to estimate the covariance.

Later, Crammer et al. [8] proposed another Perceptron-based algorithm called the Passive Aggressive (PA) algorithm [8], which uses modern margin maximization learning. The PA algorithm performs better than both the original and Second-order Perceptrons. Nevertheless, the PA algorithm only works on single view datasets. Similar algorithms that improved upon the PA algorithm include the Passive-Aggressive Mahalanobis [19], Confidence-Weight (CW) Linear

[1] http://stats.stackexchange.com/
[2] http://metaoptimize.com/qa/
[3] http://mathoverflow.net/questions

Classification [10], and CW algorithm for multi-class classification [9]. In this paper, we derive a two-view adaptive version of the PA algorithm.

3 Math Topic Classification

3.1 Math Document

Math documents contain not only textual data, but also math expressions. Math expressions embody abstract mathematical semantics via math symbols and structures. In math documents, math expressions are presented in ASCII or markup formats, e.g., LaTeX, ASCIIMath [12], OMDoc [13], OpenMath [3], and MathML [1]. Among these, the LaTeX markup language has been used by many researchers for more than 40 years. We choose LaTeX as the raw storage format for embedding math expressions in math documents, since it requires less memory compared to other markup languages. For example, the raw format of a posting from Math Overflow is given in Listing 1.1, where math expressions are enclosed between the $ symbols.

Listing 1.1. An Example Question

```
Title:   Why isn't Likelihood a Probability Density Function?
Content: I've been trying to get my head around why a likelihood isn't a
    probability density function. My understanding says that for an
    event X and a model parameter m:
        $P(X|m)$ is a probability density function
        $P(m|X)$ is not...
    It feels like it should be, and I can't find a clear explanation of
        why it's not. Does it also mean that a Likelihood can take a
        value greater than $1$?
```

Different from textual descriptions, math expressions cannot be simply encoded as an unordered sequence of tokens due to its rich math semantics, which includes functions, operators, variables, constant numbers, etc. As such, we propose a novel feature extraction method to convert math expressions into math features, as described in the following section.

3.2 Math Feature Extraction

Due to the existence of both semantic and structural information, the preprocessing step for math expressions is more complex than that of text documents. For the purpose of math topic classification, math features should be representative enough to reflect the underlying characteristics of each math topic. To do that, we perform the following steps:

- Content MathML conversion. We convert the retrieved LaTeX math expressions into Content MathML format.
- Math feature extraction. From the Content MathML data, we can extract math features by traversing the MathML tree.

To convert math expressions from LaTeX, we use the SnuggleTeX library[4]. We first convert the math expressions from LaTeX to the representation MathML format, then we use cascading stylesheets to map the representation MathML to content MathML. MathML is selected over LaTeX for its rich semantics and ease of processing via standard XML libraries. Listing 1.2 shows an example of content MathML for the math expression $(x + y)^2$.

For content MathML data, we use the XML tree traversal approach for extracting math features. In this research, we only use two kinds of features, single features and combination features. The single features are used to express constant numbers, variable names, functions names, etc. Combination features are the combinations of math operators and operands in the math expressions.

Listing 1.2. *MathML* Content Markup of $(x + y)^2$

```
<apply>                % apply operator
   <power/>            % power operator
   <mfence>
      <apply>          % apply operator
         <plus/>       % plus operator
         <ci>x</ci>    % variable (ci) x
         <ci>y</ci>    % variable (ci) y
      </apply>
   </mfence>
   <cn>2</cn>          % constant (cn) 2
</apply>
```

Take the sub-expression $x + y$ in Listing 1.2 as an example; based on the content MathML data, we have two single features (ci)x and (ci)y, where ci stands for a variable and (ci)x stands for a variable named x. We also have one combination feature (plus)(ci)x(ci)y, where plus stands for the operator + and (plus)(ci)x(ci)y denotes the operator + applied to two operands x and y.

3.3 Supervised Key Phrase Extraction

In math document classification, key phrases play a crucial role in discriminating math documents. Take "Wishart distribution" and "topological vector space" as examples, "Wishart distribution" tends to appear in documents related to statistics. On the other hand, "topological vector space" is a very common phrase in linear algebra. Therefore, in this research, we adapt the Natural Language Toolkit (NLTK)[5] to extract noun phrases from math documents. Then, the Jensen-Shannon (JS) divergence [7] is used to weight each noun phrase as follows:

$$JS(p_1, \ldots, p_C) = -(\sum_{i=1}^{C} \pi_i p_i) \log(\sum_{i=1}^{C} \pi_i p_i) + \sum_{i=1}^{C} \pi_i p_i \log(p_i) \quad (1)$$

[4] http://www2.ph.ed.ac.uk/snuggletex

[5] http://www.nltk.org/

where p_i is the probability of the phrase appearing in class i, $\sum_i \pi_i = 1$ over all C classes, and $\pi_i > 0$. JS is zero for phrases that appear uniformly over all classes, and maximized (at 1) for phrases appearing only in one class.

3.4 Online Learning Classification

Given a classification task of two classes (negative -1 and positive $+1$) and an unknown pattern represented by feature vector $\mathbf{x}$. The goal is to learn the weight $\mathbf{w}$ of a linear prediction function $f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$. The online learning algorithm operates in rounds, as input data arrives sequentially. Let $x_t \in \mathbb{R}^n$ be an example arriving at round t. The algorithm predicts its label $\hat{y}_t \in \{-1, +1\}$, after which it receives the true label. If its prediction is correct, the learning process proceeds to the next round. Otherwise, it suffers a loss $\ell(y_t, \hat{y}_t)$, and updates its weight $\mathbf{w}$ accordingly. The loss can be modeled using the hinge-loss function, which equals to zero when the margin exceeds 1, as follows.

$$\ell(\mathbf{w}_t; (\mathbf{x}_t, y_t)) = \begin{cases} 0 & \text{if} \quad y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \geq 1 \\ 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) & \text{otherwise} \end{cases} \tag{2}$$

Crammer et al. [8] formulated three optimization problems; one based on hard margin and two using soft margins, which are named PA, PA-I, and PA-II respectively, as follows.

$$\begin{aligned} \mathbf{w}_{t+1} &= \underset{\mathbf{w} \in \mathbb{R}^n}{\text{argmin}} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 \qquad \text{(PA)} \\ &\text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0. \\ \mathbf{w}_{t+1} &= \underset{\mathbf{w} \in \mathbb{R}^n}{\text{argmin}} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + C\xi \quad \text{(PA-I)} \\ &\text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi; \xi \geq 0. \\ \mathbf{w}_{t+1} &= \underset{\mathbf{w} \in \mathbb{R}^n}{\text{argmin}} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + C\xi^2 \text{ (PA-II)} \\ &\text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi. \end{aligned} \tag{3}$$

Intuitively, the new weight $\mathbf{w}_{t+1}$ should be close to the old weight $\mathbf{w}_t$ while minimizing the loss $\ell(\mathbf{w}; (\mathbf{x}_t, y_t))$. Solving the above problems, they obtained the weight update equation as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$$

where the coefficient τ_t has one of the following three forms:

$$\tau_t = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\parallel \mathbf{x}_t \parallel^2} \text{ (PA)}, \; \tau_t = \min\left\{C, \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\parallel \mathbf{x}_t \parallel^2}\right\} \text{ (PA-I), and}$$
$$\tau_t = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\parallel \mathbf{x}_t \parallel^2 + \frac{1}{2C}} \text{ (PA-II).}$$

For the two-view online learning setting, training data are triplets $(\mathbf{x}_t^A, \mathbf{x}_t^B, y_t) \in \mathbb{R}^n \times \mathbb{R}^m \times [-1, +1]$, which arrive in sequence where $\mathbf{x}_t^A \in \mathbb{R}^n$ is the first view

vector, $\mathbf{x}_t^B \in \mathbb{R}^m$ is the second view vector, and y_t is their common label. Since we don't know which view is more important than the other, the coupled weights $(\mathbf{w}_t^A, \mathbf{w}_t^B)$ should be learnt based on the weighted *hybrid model* [20] as follows:

$$f(\mathbf{x}_t^A, \mathbf{x}_t^B) = \text{sign}\Big(\eta \mathbf{w}_t^A \cdot \mathbf{x}_t^A + (1-\eta)\mathbf{w}_t^B \cdot \mathbf{x}_t^B\Big)$$

where $\eta \in (0, 1)$ is used to adjust the importance of the two views.

Let $g(\mathbf{x}_t^A, \mathbf{x}_t^B) = \eta \mathbf{w}_t^A \cdot \mathbf{x}_t^A + (1-\eta)\mathbf{w}_t^B \cdot \mathbf{x}_t^B$. To incorporate the new model into the algorithm, we define the loss function as follows:

$$\ell((\mathbf{w}_t^A, \mathbf{w}_t^B); (\mathbf{x}_t^A, \mathbf{x}_t^B, y_t)) = \begin{cases} 0 & \text{if} \quad y_t g(\mathbf{x}_t^A, \mathbf{x}_t^B) \geq 1 \\ 1 - y_t g(\mathbf{x}_t^A, \mathbf{x}_t^B) & \text{otherwise} \end{cases} \tag{4}$$

Relationship between Views. To determine the relatedness between the two views, we define a disagreement factor as follows:

$$|\eta \mathbf{w}_t^A \cdot \mathbf{x}_t^A - (1-\eta)\mathbf{w}_t^B \cdot \mathbf{x}_t^B| \tag{5}$$

where $|\cdot|$ denotes the absolute function and η, similar to the hybrid model, is used to trade off the disagreement between the two views. The objective is to minimize the disagreement between the two views.

Adaptive Two-View Passive Aggressive Algorithm. The ideal objective function should include both the new loss function in (4) and the view relatedness factor in (5). Similar to the PA algorithm, the new weights of the two-view learning algorithm are updated based on the optimization problem as follows:

$$\begin{aligned}(\mathbf{w}_{t+1}^A, \mathbf{w}_{t+1}^B) \quad = \underset{(\mathbf{w}^A, \mathbf{w}^B) \in \mathbb{R}^n \times \mathbb{R}^m}{\text{argmin}} & \frac{1}{2} \parallel \mathbf{w}^A - \mathbf{w}_t^A \parallel^2 + \frac{1}{2} \parallel \mathbf{w}^B - \mathbf{w}_t^B \parallel^2 \\ & + \gamma |\eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B| + C\xi \\ \text{s.t.} \quad 1 - y_t g(\mathbf{x}_t^A, \mathbf{x}_t^B) \leq \xi; & \quad \xi \geq 0\,.\end{aligned}$$

where γ and C are positive agreement and aggressiveness parameters respectively. While γ is used to adjust the importance of the agreement between the two views, C is used to control the aggressiveness property of the PA algorithm. Note that the multiplier y_t in the agreement is there to simplify subsequent derivations.

For the absolute function, we have

$$\begin{aligned}|\eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B| = \max \Big(& \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B, \\ & (1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B - \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A \Big)\end{aligned}$$

Suppose $z = |\eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B|$, the above optimization problem can be expressed as follows:

$$(\mathbf{w}_{t+1}^A, \mathbf{w}_{t+1}^B) = \underset{(\mathbf{w}^A, \mathbf{w}^B) \in \mathbb{R}^n \times \mathbb{R}^m}{\operatorname{argmin}} \frac{1}{2} \parallel \mathbf{w}^A - \mathbf{w}_t^A \parallel^2 + \frac{1}{2} \parallel \mathbf{w}^B - \mathbf{w}_t^B \parallel^2 + \gamma z + C\xi$$

$$\begin{aligned} \text{s.t.} \qquad & 1 - y_t g(\mathbf{x}_t^A, \mathbf{x}_t^B) \leq \xi; \quad \xi \geq 0; \\ & z \geq \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B; \\ & z \geq (1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B - \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A. \end{aligned} \tag{6}$$

Remark. When the value of the objective function is small, the disagreement factor z of the two views is forced to be small. If we set $\gamma = 0$, the problem reduces to learning two independent linear models; if we set $\gamma = 1$, we aggressively penalize any view disagreements. By adjusting the value of $0 < \gamma < 1$, we can control the amount of collaboration between the two views.

Proposition 1. *The optimization problem (6) has the following close form solution:*

$$\mathbf{w}^A = \mathbf{w}_t^A - \eta(\alpha - \beta - \tau) y_t \mathbf{x}_t^A$$

and

$$\mathbf{w}^B = \mathbf{w}_t^B - (1-\eta)(\beta - \alpha - \tau) y_t \mathbf{x}_t^B$$

where

$$\tau = \min\left\{C, \frac{(\alpha-\beta)\Big(\eta^2 \parallel \mathbf{x}_t^A \parallel^2 - (1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2\Big) + \ell_t}{\eta^2 \parallel \mathbf{x}_t^A \parallel^2 + (1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2}\right\},$$

$$\alpha = \min\left\{\gamma, \frac{1}{2}\Big(\gamma + \frac{1}{\eta}\frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\parallel \mathbf{x}_t^A \parallel^2} - \frac{1}{1-\eta}\frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\parallel \mathbf{x}_t^B \parallel^2}\Big)\right\}, \text{ and}$$

$$\beta = \min\left\{\gamma, \frac{1}{2}\Big(\gamma - \frac{1}{\eta}\frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\parallel \mathbf{x}_t^A \parallel^2} + \frac{1}{1-\eta}\frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\parallel \mathbf{x}_t^B \parallel^2}\Big)\right\}.$$

Finally, we obtain our Adaptive Two-view Passive Aggressive formulation as shown in Algorithm 1.

Getting Rid of Parameter η. One limitation of the Two-view PA algorithm in [20] is that its view parameter η must be chosen beforehand. In practice, however, choosing a suitable value for this parameter can be tedious. In addition, the optimal value may change with time, thereby affecting the performance. Therefore, we propose an adaptive variant of the Two-view PA algorithm that automatically determines the best value of η. The idea is to modify the objective function of the optimization problem (6) by adding a new regularization factor $\frac{\zeta}{2}(\eta - \eta_t)^2$. The new optimization problem has no close form expression for η since α, β, and τ all depend on η. Without loss of generality, we assume that these variables only depend on the *previous* value of η, i.e., η_t.

Proposition 2. *Suppose that α, β, and τ are independent of η, the new optimization problem has an approximated close form solution as follows.*

$$\eta = \eta_t - \frac{1}{\zeta}\Big((\alpha - \beta - \tau) y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (\beta - \alpha - \tau) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B\Big) \tag{7}$$

Initially the two views are treated equally, i.e., $\eta = 0.5$, and will be updated based on Equation (7) thereafter. This is thus an adaptive version of the Two-view PA algorithm, which we call the Adaptive Two-view PA algorithm.

Algorithm 1. Adaptive Two-view Passive Aggressive Algorithm

Input:
C = positive aggressiveness parameter
γ = positive agreement parameter
Output:
None
Process:
Initialize $\mathbf{w}_1^A \leftarrow \mathbf{0}$; $\mathbf{w}_1^B \leftarrow \mathbf{0}$; $\eta = 0.5$;
for $t = 1, 2, \ldots$ **do**
Receive instances $\mathbf{x}_t^A \in \mathbb{R}^n$ and $\mathbf{x}_t^B \in \mathbb{R}^m$
Predict $\hat{y}_t = \text{sign}(\eta \mathbf{w}_t^A \cdot \mathbf{x}_t^A + (1-\eta)\mathbf{w}_t^B \cdot \mathbf{x}_t^B)$
Receive correct label $y_t \in \{-1, +1\}$
Suffer loss
$\ell_t \leftarrow \max\left\{0, 1 - \eta y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B\right\}$
if $\ell_t > 0$ **then**
Update $\mathbf{w}_t^A$ and $\mathbf{w}_t^B$ per Proposition 1
Update η per Proposition 2
end if
end for

Table 1. Summary of Datasets in Our Experiments

	View		Sample Count		
	Name	#Dim	#Pos	#Neg	#Total
Ads	img & dest	929	459	2820	3279
	alt & base	602			
Product Review	lexical	2759	1000	1000	2000
	formal	5			
WebKB	page	3000	230	821	1051
	link	1840			

4 Performance Evaluation

In this section, we evaluate the online classification performance of our proposed Adaptive Two-view PA algorithm on three benchmark datasets, Ads [14], Product Review [17], and WebKB [23], and a math dataset taken from the Math Overflow site. The *single-view* PA algorithm serves as the baseline. We employ a different PA model for each view, denoted as *PA View 1* and *PA View 2* for each view, respectively. We also report the results from a simple concatenation of the input feature vectors from each view to form a larger feature set, denoted

as *PA Cat*. We compare the performance of all algorithms based on F-measure instead of accuracy because most of the datasets are highly (class) imbalanced.

4.1 Two-View Learning Evaluation

In this section, we evaluate the proposed algorithm on three benchmark datasets. The dataset characteristics are listed in Table 1. We use cross validation to select the optimal value for all parameters C, γ, and η, so as to make the comparison fair and meaningful. Here, η is learnt using the Adaptive Two-view PA algorithm.

Table 2. F-Measure on 3 benchmark datasets

Dataset	PA View 1	PA View 2	PA Cat	Adaptive Two-view PA
Ads	83.41 ± 2.76	76.26 ± 2.21	81.95 ± 2.55	$\mathbf{84.96} \pm 2.41$
Product Review	86.69 ± 1.39	70.80 ± 1.57	86.68 ± 1.63	$\mathbf{88.72} \pm 1.80$
WebKB	93.76 ± 2.30	90.68 ± 2.70	95.56 ± 0.98	$\mathbf{98.02} \pm 2.14$

The Ads dataset was first used by Kushmerick [14] to automatically filter advertisement images from web pages. In our experiments, we used just four views, namely *image URL view*, *destination URL view*, *base URL view*, and *alt view*. Since we are limited to handling two views for each task, the first and second views were concatenated into View 1 and the remaining two views were concatenated into View 2. This dataset has 3279 examples, including 459 positive examples (ads), with the remaining samples negative (non-ads). Experimental learning results on the Ads dataset are shown in Table 2, which shows the proposed algorithm to have the best F-measure score, performing more than 1% better than the runner-up, PA View 1. Interestingly, PA Cat fared worse than PA View 1, which could be due to a noisy decision boundary in the space of PA View 2. This can also be seen by the marginal improvement of 1% of the adaptive Two-view PA results.

The Product Review dataset is crawled from popular online Chinese cellphone forums [17]. The dataset has 1000 true reviews and 1000 spam reviews. It comprises two sets of features: one based on review content (*lexical view*) and the other based on extracted characteristics of the review sentences (*formal view*). The experimental results on this dataset are shown in Table 2. Again, our Two-view adaptive PA performs better than the rest, beating the runner-up (PA View 1) by 2%. Here PA Cat performed better than either view alone, which is typically the case.

The WebKB course dataset has been frequently used in the empirical study of multi-view learning. It comprises 1051 web pages collected from the computer science departments of four universities. Each page has a class label, course or non-course. The two views of each page are the textual content of a web page (*page view*) and the words that occur in the hyperlinks of other web pages pointing to it (*link view*), respectively. We used a processed version of the WebKB

course dataset [23] in our experiment. The performance of PA Cat here is also better than the best single-view PA. And the Adaptive Two-view PA performed more than 2% better than PA Cat, and 4% better than the best individual view PA. Compared to the non-adaptive (equal weightage of both views) Two-view approach of [20], our adaptive Two-view approach performed similarly or better.

4.2 Math Topic Classification

We downloaded more than 30,000 math questions and answers belonging to 20 math categories such as algebraic geometry, number theory, algebraic topology, combinatorics, group theory, probability, etc., from the Math Overflow site, a popular math question answering system. Each math question or answer is treated as one math document, which may contain both text content and math expressions. To evaluate math topic classification, we choose two major categories (algebraic geometry and number theory), which comprises more than 7,200 math documents. After preprocessing, we obtain the following datasets.

- Text only. All math expressions are removed from math documents. The remaining text-only documents are transformed into a vector format using $tf \times idf$ weighting [18].
- Math only. To evaluate whether math expressions are useful for math topic classification, we extract all math expressions from each math document. Then we apply the math feature extraction method of Section 3.2 to generate the *math only* dataset.
- Raw. It is trivial to treat math expressions as normal text data. One can use the latest text preprocessing techniques to extract textual features from both math and text.
- Math and text. We store the *math only* and *text only* datasets as two datasets (views), called the *math & text* dataset.
- Key phrase. Math key phrases extracted using the method of Section 3.3.

PA Only. After preprocessing, we then run the PA algorithm on all datasets using 5-fold cross validation. The experimental results are shown in column 2 of Table 3. Note that the PA model trained on Math & Text operates on a concatenation of the two views. We see that the PA algorithm performed the worst on the *math only* dataset and best on the *text only* dataset. Clearly, there is much room for improvement in our math expression extraction process.

For the *raw text* and *math & text* datasets, the F-measure of the PA algorithm is not high, although both math and text data are taken into consideration. In fact, the PA algorithm did very poorly on just the *raw text*. Its performance is only better than its *math only* dataset results. Compared with the *math only* and *raw text* datasets, the *math & text* dataset can improve the performance of the PA algorithm. However, its performance on this dataset is actually worse than the *text only* dataset.

The Missing View. In practice, math documents do not always contain both text data and math expressions. So what happens if either text data or math expressions are available, but not both? Can Two-view PA work in this case? To find out, we trained the Two-view PA on the *text & math* dataset and tested it on the *text only* and *math only* datasets. It means that we trained the Two-view PA on both views to have two weight vectors and then used them to predict the labels for documents in individual views. While testing on one view, we will ignore the other view.

Table 3. F-Measure Comparison on Math Overflow Datasets (* trained on the Math & Text Dataset, with results for individual views shown)

Dataset	PA	Adaptive Two-view PA*
Text Only	72.73 ± 2.97	$\mathbf{73.85} \pm 2.90$
Math Only	56.31 ± 7.47	$\mathbf{64.25} \pm 6.05$
Raw	61.02 ± 0.12	-
Math & Text	68.91 ± 5.03	$\mathbf{75.70} \pm 3.37$
Key Phrase	76.78 ± 0.90	$\mathbf{78.15} \pm 1.27$

We also ran the Two-view PA algorithm on the *math & text* dataset (treating each as one view), whose 75.70% F-measure score is more than 6% better than the PA algorithm (68.91%), which was trained on the combined view. Moreover, compared with the PA on *text only* dataset (72.73%), the two view performance (75.70%) is improved by nearly 3%.

This means when user posts a math question containing math expressions but without text data, the Two-view PA algorithm performs better than the PA algorithm trained on the *math only* dataset by up to 6%. The results are encouraging because we are able to take advantage of the data of one view to improve the performance of the classifier on another view by enforcing agreement between the two views.

Similarly, for the *key phrase* dataset, the performance is improved by up to 4% compared to the *text only* dataset in the single view PA. If we train the Two-view PA with the *key phrase* and *math only* datasets, the Two-view performs better than the single view PA by nearly 2%.

Table 4. The Adaptive Two-view PA Results for All Datasets

Dataset	η	F-measure
Ads	0.342 ± 0.015	84.96 ± 2.41
Math Overflow	0.524 ± 0.001	75.70 ± 3.37
Product Review	0.795 ± 0.011	88.72 ± 1.80
WebKB	0.438 ± 0.001	98.02 ± 2.14

4.3 View Weight Parameter Learning

The average η value for the Adaptive Two-view PA algorithm for all datasets are listed in Table 4. Note that for all 3 datasets, View 1 performed better than View 2, individually. For the Ads and WebKB datasets, we note that $\eta < 0.5$ despite View 1 performing better than View 2. On the other hand, for the Math Overflow and Product Review datasets, we have $\eta > 0.5$. Therefore, we cannot rely solely on performance of individual views to determine the value of η. Generally, the better performing view does not automatically deserves a higher weightage. The Adaptive Two-view PA algorithm can solve this problem by adaptively updating η at each round of the learning process.

5 Conclusion

We proposed an Adaptive Two-view Passive Aggressive algorithm, which is able to take advantage of multiple views of data to achieve an improvement in overall classification performance. We formulated our learning framework into an optimization problem and derive a closed form solution. Making a simple assumption on the independence of other parameters on the weight parameter, we derived an approximated close-form update equation for the view mixing parameter.

We evaluated the proposed approach on practical applications such as product review classification, advertising image removal, and math topic classification. We also prepared a two-view Math Overflow dataset containing text and math expressions, which is useful for math topic classification since this is the first publicly available dataset of its kind.

Although in this research, we evaluated the proposed approach on math questions and answers, it can be applied in practice to deal with other kinds of math documents such as math questions in student books, scientific papers, etc. There remain some interesting open problems that warrant further investigations. First, the math feature extraction method should be investigated further because the performance based on math features only is not good enough. We would also like to extend the Two-view PA algorithm to handle multiple views and multiple classes. However, formulating a multi-view PA is non-trivial, as it involves defining multi-view relatedness and minimizing pairs of view agreements. Formulating a multi-class Two-view PA should be more feasible.

Acknowledgements. We thank the anonymous reviewers for their valuable comments and suggestions, especially one reviewer who pointed out the flaw in the original presentation of proposition 2. This research was supported in part by Singapore Ministry of Education's Academic Research Fund Tier 2 grant ARC 9/12 (MOE2011-T2-2-056).

References

1. Ausbrooks, R., Buswell, S., Dalmas, S., Devitt, S., Diaz, A., Hunter, R., Smith, B., Soiffer, N., Sutor, R., Watt, S.: Mathematical markup language (mathml) version 2.0 (2000)

2. Block, H.: The perceptron: A model for brain functioning. Rev. Modern Phys. 34, 123–135 (1962)
3. Buswell, S., Caprotti, O., Carlisle, D.P., Dewar, M.C., Gaetano, M., Kohlhase, M.: The Open Math standard version 2.0 (2004)
4. Cesa-Bianchi, N., Conconi, A., Gentile, C.: A second-order perceptron algorithm. Siam J. of Comm. 34 (2005)
5. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20, 273–297 (1995)
6. Cover, T., Hart, P.: Nearest Neighbor Pattern Classification 13, 373–389 (2002)
7. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley-Interscience, New York (1991)
8. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research, 551–585 (2006)
9. Crammer, K., Dredze, M., Kulesza, A.: Multi-class confidence weighted algorithms. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Singapore, pp. 496–504. Association for Computational Linguistics,
10. Dredze, M., Crammer, K., Pereira, F.: Confidence-weighted linear classification. In: ICML 2008: Proceedings of the 25th International Conference on Machine Learning, pp. 264–271. ACM, New York (2008)
11. Farquhar, J.D.R., Hardoon, D.R., Meng, H., Shawe-Taylor, J., Szedmák, S.: Two view learning: Svm-2k, theory and practice. In: Proceedings of NIPS 2005 (2005)
12. Jipsen, P.: Translating ascii math notation to mathml and graphics (2007)
13. Kohlhase, M., Sucan, I.: A Search Engine for Mathematical Formulae. In: Calmet, J., Ida, T., Wang, D. (eds.) AISC 2006. LNCS (LNAI), vol. 4120, pp. 241–253. Springer, Heidelberg (2006)
14. Kushmerick, N.: Learning to remove internet advertisements. In: Proceedings of the Third Annual Conference on Autonomous Agents, AGENTS 1999, pp. 175–181. ACM, New York (1999)
15. Langley, P., Iba, W., Thompson, K.: An analysis of bayesian classifiers. In: AAAI 1992: Proceedings of the Tenth National Conference on Artificial Intelligence, pp. 223–228. AAAI Press (1992)
16. Lewis, D.D.: Naive (bayes) at Forty: The Independence Assumption in Information Retrieval. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 4–15. Springer, Heidelberg (1998)
17. Li, G., Hoi, S.C.H., Chang, K.: Two-view transductive support vector machines. In: Proceedings of SDM 2010, pp. 235–244 (2010)
18. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
19. Nguyen, T.T., Chang, K., Hui, S.C.: Distribution-aware online classifiers. In: Walsh, T. (ed.) IJCAI, pp. 1427–1432. IJCAI/AAAI (2011)
20. Nguyen, T.T., Chang, K., Hui, S.C.: Two-View Online Learning. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part I. LNCS, vol. 7301, pp. 74–85. Springer, Heidelberg (2012)
21. Novikoff, A.: On convergence proofs of perceptrons. In: Proceedings of the Symposium on the Mathematical Theory of Automata, vol. 7, pp. 615–622 (1962)
22. Quinlan, J.R., Rivest, R.L.: Inferring decision trees using the minimum description length principle. Inf. Comput. 80(3), 227–248 (1989)
23. Sindhwani, V., Niyogi, P., Belkin, M.: Beyond the point cloud: from transductive to semi-supervised learning. In: Proceedings of the 22nd International Conference on Machine Learning, ICML 2005, pp. 824–831. ACM, New York (2005)

A Proof of Proposition 1

To prove the Proposition 1, we first define the Lagrangian of the optimization problem as follows:

$$\begin{aligned}\mathcal{L} &= \frac{1}{2} \parallel \mathbf{w}^A - \mathbf{w}_t^A \parallel^2 + \frac{1}{2} \parallel \mathbf{w}^B - \mathbf{w}_t^B \parallel^2 + \gamma z + C\xi - \lambda\xi \\ &\quad + \tau\Big(1 - \xi - \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B\Big) \\ &\quad + \alpha\Big(\eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B - z\Big) \\ &\quad + \beta\Big((1-\eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B - \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - z\Big) \\ &= \frac{1}{2} \parallel \mathbf{w}^A - \mathbf{w}_t^A \parallel^2 + \frac{1}{2} \parallel \mathbf{w}^B - \mathbf{w}_t^B \parallel^2 + (\gamma - \alpha - \beta) z + (C - \lambda - \tau)\xi \\ &\quad + \eta(\alpha - \beta - \tau) y_t \mathbf{w}^A \cdot \mathbf{x}_t^A + (1-\eta)(\beta - \alpha - \tau) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B + \tau\end{aligned} \tag{8}$$

where α, β, τ, and λ are positive Lagrangian multipliers.

Setting the partial derivatives of $\mathcal{L}$ with respect to the weight $\mathbf{w}^A$ to zero, we have

$$0 = \frac{\partial \mathcal{L}}{\partial \mathbf{w}^A} = \mathbf{w}^A - \mathbf{w}_t^A + \eta(\alpha - \beta - \tau) y_t \mathbf{x}_t^A \Rightarrow \mathbf{w}^A = \mathbf{w}_t^A - \eta(\alpha - \beta - \tau) y_t \mathbf{x}_t^A \tag{9}$$

Similarly, for the other view we have

$$\mathbf{w}^B = \mathbf{w}_t^B - (1-\eta)(\beta - \alpha - \tau) y_t \mathbf{x}_t^B \tag{10}$$

Setting the partial derivatives of $\mathcal{L}$ with respect to weight z to zero, we have

$$0 = \frac{\partial \mathcal{L}}{\partial z} = (\gamma - \alpha - \beta) \Rightarrow \alpha + \beta = \gamma \tag{11}$$

Setting the partial derivatives of $\mathcal{L}$ with respect to weight ξ to zero, we have

$$0 = \frac{\partial \mathcal{L}}{\partial \xi} = (C - \lambda - \tau) \Rightarrow \lambda + \tau = C \tag{12}$$

Note that $\lambda \geq 0$, so we can conclude that $0 \leq \tau \leq C$.

Substituting (9), (10), (11), and (12) into (8), we have

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\eta^2(\alpha - \beta - \tau)^2 \parallel \mathbf{x}_t^A \parallel^2 + \frac{1}{2}(1-\eta)^2(\beta - \alpha - \tau)^2 \parallel \mathbf{x}_t^B \parallel^2 \\ &\quad + \eta(\alpha - \beta - \tau) y_t \Big(\mathbf{w}_t^A - \eta(\alpha - \beta - \tau) y_t \mathbf{x}_t^A\Big) \cdot \mathbf{x}_t^A \\ &\quad + (1-\eta)(\beta - \alpha - \tau) y_t \Big(\mathbf{w}_t^B - (1-\eta)(\beta - \alpha - \tau) y_t \mathbf{x}_t^B\Big) \cdot \mathbf{x}_t^B + \tau \\ &= -\frac{1}{2}\eta^2(\alpha - \beta - \tau)^2 \parallel \mathbf{x}_t^A \parallel^2 - \frac{1}{2}(1-\eta)^2(\beta - \alpha - \tau)^2 \parallel \mathbf{x}_t^B \parallel^2 \\ &\quad + \eta(\alpha - \beta - \tau) y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A + (1-\eta)(\beta - \alpha - \tau) y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B + \tau\end{aligned} \tag{13}$$

Setting the partial derivatives of $\mathcal{L}$ with respect to weight τ to zero, we have

$$\begin{aligned} 0 = \frac{\partial \mathcal{L}}{\partial \tau} &= \eta^2(\alpha - \beta - \tau) \parallel \mathbf{x}_t^A \parallel^2 + (1-\eta)^2(\beta - \alpha - \tau) \parallel \mathbf{x}_t^B \parallel^2 \\ &\quad + 1 - \eta y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B) \\ \Rightarrow \tau &= \frac{(\alpha - \beta)\Big(\eta^2 \parallel \mathbf{x}_t^A \parallel^2 - (1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2 \Big) + \ell_t}{\eta^2 \parallel \mathbf{x}_t^A \parallel^2 + (1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2} \end{aligned}$$

where the loss $\ell_t = 1 - \eta y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A - (1-\eta) y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B$. For the sake of simplicity, we denote

$$a = \frac{1}{\eta^2 \parallel \mathbf{x}_t^A \parallel^2 + (1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2} \quad \text{and} \quad b = \parallel \mathbf{x}_t^A \parallel^2 \parallel \mathbf{x}_t^B \parallel^2 \tag{14}$$

As mentioned in Equation (12), we have $\tau + \lambda = C$ and $\lambda \geq 0$, we can conclude that $\tau \leq C$. Now τ can be determined as follows:

$$\tau = \min \Big\{ C, a\Big((\alpha - \beta)(\eta^2 \parallel \mathbf{x}_t^A \parallel^2 - (1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2) + \ell_t \Big) \Big\} \tag{15}$$

Substituting (15) into (13), we have

$$\begin{aligned} \mathcal{L} = &-\frac{1}{2} a^2 \eta^2 \Big((\alpha - \beta)(1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2 - \ell_t \Big)^2 \parallel \mathbf{x}_t^A \parallel^2 \\ &-\frac{1}{2} a^2 (1-\eta)^2 \Big((\beta - \alpha)\eta^2 \parallel \mathbf{x}_t^A \parallel^2 - \ell_t \Big)^2 \parallel \mathbf{x}_t^B \parallel^2 \\ &+ a\eta((\alpha - \beta)(1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2 - \ell_t) y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A \\ &+ a(1-\eta)((\beta - \alpha)\eta^2 \parallel \mathbf{x}_t^A \parallel^2 - \ell_t) y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B \\ &+ a\Big((\alpha - \beta)(\eta^2 \parallel \mathbf{x}_t^A \parallel^2 - (1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2) + \ell_t \Big) \end{aligned} \tag{16}$$

Setting the partial derivatives of $\mathcal{L}$ with respect to weight α to zero, we have

$$\begin{aligned} 0 = \frac{\partial \mathcal{L}}{\partial \alpha} &= -a^2 \eta^2 \Big((\alpha - \beta)(1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2 - \ell_t \Big) b \\ &\quad + a^2 (1-\eta)^2 \Big((\beta - \alpha)\eta^2 \parallel \mathbf{x}_t^A \parallel^2 - \ell_t \Big) b \\ &\quad + a\eta(1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2 y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A - a(1-\eta)\eta^2 \parallel \mathbf{x}_t^A \parallel^2 y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B \\ &\quad + a(\eta^2 \parallel \mathbf{x}_t^A \parallel^2 - (1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2) \end{aligned} \tag{17}$$

Simplifying the above equality, we have

$$\begin{aligned} &-b\eta(1-\eta)(\alpha - \beta)(\eta^2 \parallel \mathbf{x}_t^A \parallel^2 + (1-\eta)^2 \parallel \mathbf{x}_t^B \parallel^2) \\ &+ (1-\eta) \parallel \mathbf{x}_t^B \parallel^2 y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A - \eta \parallel \mathbf{x}_t^A \parallel^2 y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B = 0 \end{aligned} \tag{18}$$

Hence, we have

$$\alpha - \beta = \frac{1}{\eta} \frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\parallel \mathbf{x}_t^A \parallel^2} - \frac{1}{1-\eta} \frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\parallel \mathbf{x}_t^B \parallel^2}$$

Recall that we have $\alpha + \beta = \gamma$. Therefore, we can conclude that

$$\alpha = \frac{1}{2}\Big(\gamma + \frac{1}{\eta}\frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\| \mathbf{x}_t^A \|^2} - \frac{1}{1-\eta}\frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\| \mathbf{x}_t^B \|^2}\Big) \tag{19}$$

Similarly, we have

$$\beta = \frac{1}{2}\Big(\gamma - \frac{1}{\eta}\frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\| \mathbf{x}_t^A \|^2} + \frac{1}{1-\eta}\frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\| \mathbf{x}_t^B \|^2}\Big) \tag{20}$$

Recall that we have $\alpha \geq 0$, $\beta \geq 0$, and $\alpha + \beta = \gamma$. Hence, we can conclude that $0 \leq \alpha \leq \gamma$ and $0 \leq \beta \leq \gamma$. That is

$$\alpha = \min\Big\{\gamma, \frac{1}{2}\Big(\gamma + \frac{1}{\eta}\frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\| \mathbf{x}_t^A \|^2} - \frac{1}{1-\eta}\frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\| \mathbf{x}_t^B \|^2}\Big)\Big\}$$
$$\beta = \min\Big\{\gamma, \frac{1}{2}\Big(\gamma - \frac{1}{\eta}\frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\| \mathbf{x}_t^A \|^2} + \frac{1}{1-\eta}\frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\| \mathbf{x}_t^B \|^2}\Big)\Big\}$$

B Proof of Proposition 2

For the Adaptive PA algorithm, we have new Lagrangian as follows:

$$\begin{aligned}\mathcal{L} = &\frac{1}{2} \| \mathbf{w}^A - \mathbf{w}_t^A \|^2 + \frac{1}{2} \| \mathbf{w}^B - \mathbf{w}_t^B \|^2 \\ &+(\gamma - \alpha - \beta)z + (C - \lambda - \tau)\xi + \eta(\alpha - \beta - \tau) y_t \mathbf{w}^A \cdot \mathbf{x}_t^A \\ &+(1-\eta)(\beta - \alpha - \tau) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B + \tau + \frac{\zeta}{2}(\eta - \eta_t)^2\end{aligned} \tag{21}$$

Assuming that α, β, and τ are independent on the new value of η, we have $\frac{\partial \alpha}{\partial \eta} = 0$, $\frac{\partial \beta}{\partial \eta} = 0$, and $\frac{\partial \tau}{\partial \eta} = 0$.

Setting the partial derivatives of $\mathcal{L}$ with respect to the variable η to zero, we have

$$\begin{aligned}0 = \frac{\partial \mathcal{L}}{\partial \eta} &= \zeta(\eta - \eta_t) + (\alpha - \beta - \tau) y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (\beta - \alpha - \tau) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B \\ \Rightarrow \eta &= \eta_t - \frac{1}{\zeta}\Big((\alpha - \beta - \tau) y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (\beta - \alpha - \tau) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B\Big)\end{aligned} \tag{22}$$

BDUOL: Double Updating Online Learning on a Fixed Budget

Peilin Zhao and Steven C.H. Hoi

School of Computer Engineering,
Nanyang Technological University, Singapore
{zhao0106,chhoi}@ntu.edu.sg

Abstract. Kernel-based online learning often exhibits promising empirical performance for various applications according to previous studies. However, it often suffers a main shortcoming, that is, the unbounded number of support vectors, making it unsuitable for handling large-scale datasets. In this paper, we investigate the problem of budget kernel-based online learning that aims to constrain the number of support vectors by a predefined budget when learning the kernel-based prediction function in the online learning process. Unlike the existing studies, we present a new framework of budget kernel-based online learning based on a recently proposed online learning method called "Double Updating Online Learning" (DUOL), which has shown state-of-the-art performance as compared with the other traditional kernel-based online learning algorithms. We analyze the theoretical underpinning of the proposed Budget Double Updating Online Learning (BDUOL) framework, and then propose several BDUOL algorithms by designing different budget maintenance strategies. We evaluate the empirical performance of the proposed BDUOL algorithms by comparing them with several well-known budget kernel-based online learning algorithms, in which encouraging results validate the efficacy of the proposed technique.

1 Introduction

The goal of kernel-based online learning is to incrementally learn a nonlinear kernel-based prediction function from a sequence of training instances [1–4]. Although it often yields significantly better performance than linear online learning, the main shortcoming of kernel-based online learning is its potentially unbounded number of support vectors with the kernel-based prediction function, which thus requires a large amount of memory for storing support vectors and a high computational cost of making predictions at each iteration, making it unsuitable for large-scale applications. In this paper, we aim to tackle this challenge by studying a framework for kernel-based online learning on a fixed budget or known as "budget online learning" for short, in which the number of support vectors for the prediction function is bounded by some predefined budget size.

In literature, several algorithms have been proposed for budget online learning. Crammer et al. [5] proposed a heuristic approach for budget online learning by

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 810–826, 2012.

extending the classical kernel-based perceptron method [6], which was further improved in [7]. The basic idea of these two algorithms is to remove the support vector that has the least impact on the classification performance whenever the budget, i.e., the maximal number of support vectors, is reached. The main shortcoming of these two algorithms is that they are heuristic without solid theoretic supports (e.g., no any mistake/regret bound was given).

Forgetron [8] is perhaps the first approach for budget online learning that offers a theoretical bound of the total number of mistakes. In particular, at each iteration, if the online classifier makes a mistake, it conducts a three-step updating: (i) it first runs the standard Perceptron [6] for updating the prediction function; (ii) it then shrinks the weights of support vectors by a carefully chosen scaling factor; and (iii) it finally removes the support vector with the least weight. Another similar approach is the Randomized Budget Perceptron (RBP) [9], which randomly removes one of existing support vectors when the number of support vectors exceeds the predefined budget. In general, RBP achieves similar mistake bound and empirical performance as Forgetron.

Unlike the above strategy that discards one support vector to maintain the budget, Projectron [10] adopts a projection strategy to bound the number of support vectors. Specifically, at each iteration where a training example is misclassified, it first updates the kernel classifier by applying a standard Perceptron; it then projects the new classifier into the space spanned by all the support vectors except the new example received at the current iteration, if the difference between the new classifier and its projection is less than a given threshold, otherwise it will remain unchanged. Empirical studies show that Projectron usually outperforms Forgetron in classification but with significantly longer running time. In addition to its high computational cost, another shortcoming of Projectron is that although the number of support vectors is bounded, it is unclear the exact number of support vectors achieved by Projectron in theory.

The above budget online learning approaches were designed based on the Perceptron learning framework [6]. In this paper, we propose a new framework of Budget Double Updating Online Learning (BDUOL) based on a recently proposed Double Updating Online Learning (DUOL) technique [4], which has shown state-of-the-art performance for online learning. The key challenge is to develop an appropriate strategy for maintaining the budget whenever the size of support vectors overflows. In this paper, following the theory of double updating online learning, we analyze the theoretical underpinning of the BDUOL framework, and propose a principled approach to developing three different budget maintenance strategies. We also analyze the mistake bounds of the proposed BDUOL algorithms and evaluate their empirical performance extensively.

The rest of the paper is organized as follows. Section 2 first introduces the problem setting and then presents both theoretical and algorithmic framework of the proposed budget online learning technique. Section 3 presents several different budget maintenance strategies for BDUOL. Section 4 discusses our empirical studies. Section 5 concludes this work.

2 Double Updating Online Learning on a Fixed Budget

In this section, we first introduce the problem setting for online learning and Double Updating Online Learning (DUOL), and then present the details of the proposed Budget Double Updating Online Learning framework.

2.1 Problem Setting

We consider the problem of online classification on a fixed budget. Our goal is to learn a prediction function $f : \mathbb{R}^d \to \mathbb{R}$ from a sequence of training examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$, where $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional instance and $y_t \in \mathcal{Y} = \{-1, +1\}$ is the class label assigned to $\mathbf{x}_t$. We use $sign(f(\mathbf{x}))$ to predict the class assignment for any $\mathbf{x}$, and $|f(\mathbf{x})|$ to measure the classification confidence. Let $\ell(f(\mathbf{x}), y) : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ be the loss function that penalizes the deviation of estimates $f(\mathbf{x})$ from observed labels y. We refer to the output f of the learning algorithm as a *hypothesis* and denote the set of all possible hypotheses by $\mathcal{H} = \{f | f : \mathbb{R}^d \to \mathbb{R}\}$.

In this paper, we consider $\mathcal{H}$ a Reproducing Kernel Hilbert Space (**RKHS**) endowed with a kernel function $\kappa(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ [11] implementing the inner product$\langle \cdot, \cdot \rangle$ such that: 1) reproducing property $\langle f, \kappa(\mathbf{x}, \cdot) \rangle = f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^d$; 2) $\mathcal{H}$ is the closure of the span of all $\kappa(\mathbf{x}, \cdot)$ with $\mathbf{x} \in \mathbb{R}^d$, that is, $\kappa(\mathbf{x}, \cdot) \in \mathcal{H}$ for every $\mathbf{x} \in \mathcal{X}$. The inner product $\langle \cdot, \cdot \rangle$ induces a norm on $f \in H$ in the usual way: $\|f\|_{\mathcal{H}} := \langle f, f \rangle^{\frac{1}{2}}$. To make it clear, we use $\mathcal{H}_\kappa$ to denote an RKHS with explicit dependence on kernel function κ. Throughout the analysis, we assume $\kappa(\mathbf{x}, \mathbf{x}) \leq 1$ for any $\mathbf{x} \in \mathbb{R}^d$.

2.2 Double Updating Online Learning: A Review

Our BDUOL algorithm is designed based on the state-of-the-art Double Updating Online Learning (DUOL) method [4]. Unlike traditional online learning algorithms that usually perform a single update for each misclassified example, DUOL not only updates the weight for the newly added Support Vector (SV), but also updates that of another existing SV, which conflicts most with the new SV. Furthermore, both theoretical and empirical analysis have demonstrated the effectiveness of this algorithm. Below we briefly review the basics of double updating online learning.

Consider an incoming instance $\mathbf{x}_t$ received at the t-th step of online learning. The algorithm predicts the class label $\widehat{y}_t = \text{sgn}(f_{t-1}(\mathbf{x}_t))$ using the following kernel-based classifier:

$$f_{t-1}(\cdot) = \sum_{i \in S_{t-1}} \widehat{\gamma}_i y_i \kappa(\mathbf{x}_i, \cdot),$$

where S_{t-1} is the index set of the SVs for the $(t-1)$-th step, and $\widehat{\gamma}_i$ is the weight of the i-th existing support vector. After making the prediction, the algorithm will suffer a loss, defined by a hinge loss as $\ell(f_{t-1}(\mathbf{x}_t), y_t) = \max(0, 1 - y_t f_{t-1}(\mathbf{x}_t))$.

If $\ell(f_{t-1}(\mathbf{x}_t), y_t) > 0$, the DUOL algorithm will update the prediction function f_{t-1} to f_t by adding the training example $(\mathbf{x}_t, y_t)$ as a new support vector.

Specifically, when the new added example $(\mathbf{x}_t, y_t)$ conflicts with $(\mathbf{x}_b, y_b)$, $b \in S_{t-1}$, by satisfying conditions: 1)$\ell_t = 1 - y_t f_{t-1}(\mathbf{x}_t) > 0$; 2) $\ell_b = 1 - y_b f_{t-1}(\mathbf{x}_b) > 0$; 3) $y_t y_b \kappa(\mathbf{x}_t, \mathbf{x}_b) \leq \min(-\rho, y_t y_a \kappa(\mathbf{x}_t, \mathbf{x}_a))$, $a \in S_{t-1}$, and $a \neq b$, where $\rho \in [0, 1)$ is a threshold, then the updating strategy referred as double updating will be adopted as follows:

$$f_t(\cdot) = f_{t-1}(\cdot) + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot) + d_{\gamma_b} y_b \kappa(\mathbf{x}_b, \cdot),$$

where γ_t and d_{γ_b} are computed in the following equations:

$$(\gamma_t, d_{\gamma_b}) = \begin{cases} (C, C - \hat{\gamma}_b) & \text{if } (k_t C + w_{ab}(C - \hat{\gamma}_b) - \ell_t) < 0 \text{ and} \\ & \quad (k_b(C - \hat{\gamma}_b) + w_{ab} C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab} C}{k_b}) & \text{if } \frac{w_{ab}^2 C - w_{ab}\ell_b - k_t k_b C + k_b \ell_t}{k_b} > 0 \text{ and} \\ & \quad \frac{\ell_b - w_{ab} C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \\ (\frac{\ell_t - w_{ab}(C - \hat{\gamma}_b)}{k_t}, C - \hat{\gamma}_b) & \text{if } \frac{\ell_t - w_{ab}(C - \hat{\gamma}_b)}{k_t} \in [0, C] \text{ and} \\ & \quad \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab}\frac{\ell_t - w_{ab}(C - \hat{\gamma}_b)}{k_t} > 0 \\ (\frac{k_b \ell_t - w_{ab}\ell_b}{k_t k_b - w_{ab}^2}, \frac{k_t \ell_b - w_{ab}\ell_t}{k_t k_b - w_{ab}^2}) & \text{if } \frac{k_b \ell_t - w_{ab}\ell_b}{k_t k_b - w_{ab}^2} \in [0, C] \text{ and} \\ & \quad \frac{k_t \ell_b - w_{ab}\ell_t}{k_t k_b - w_{ab}^2} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \end{cases}, \quad (1)$$

where $k_t = \kappa(\mathbf{x}_t, \mathbf{x}_t)$, $k_b = \kappa(\mathbf{x}_b, \mathbf{x}_b)$, $w_{ab} = y_t y_b \kappa(\mathbf{x}_t, \mathbf{x}_b)$ and $C > 0$; or when no existing SV conflicts with $(\mathbf{x}_t, y_t)$, the single update strategy will be adopted as follows:

$$f_t(\cdot) = f_{t-1}(\cdot) + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot), \quad (2)$$

where $\gamma_t = \min(C, \ell_t / k_t^2)$. It is not difficult to see that the single update strategy is reduced to the Passive-Aggressive updating strategy [3].

2.3 Framework of Budget Double Updating Online Learning

Although DUOL outperforms various traditional single updating algorithms, one major limitation is that it does not bound the number of support vectors, which could result in high computation and heavy memory cost when being applied to large-scale applications. In this paper, we aim to overcome this limitation by proposing a budget double updating online learning framework in which the number of support vectors is bounded by a predefined budget size.

Let us denote by B a predefined budget size for the maximal number of support vectors associated with the prediction function. The key difference of Budget DUOL over regular DUOL is to develop an appropriate budget maintenance step to ensure that the number of support vectors with the classifier f_t is less than the budget B at the beginning of each online updating step. In particular, let us denote by f_{t-1} the classifier produced by a regular DUOL at the $t-1$-th step, when the support vector size of f_{t-1} is equal to the B, BDUOL performs the classifier update towards budget maintenance: $f_{t-1} \leftarrow f_{t-1} - \Delta f_{t-1}$ such that

Algorithm 1. The Budget Double Updating Online Learning Algorithm (**BDUOL**)

PROCEDURE
Initialize $S_0 = \emptyset$, $f_0 = 0$;
for t=1,2,. . . ,T **do**
  Receive a new instance $\mathbf{x}_t$;
  Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$;
  Receive its label y_t;
  $\ell_t = max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$;
  if $\ell_t > 0$ **then**
    if ($|S_t| == B$) **then**
      $f_{t-1} = f_{t-1} - \Delta f_{t-1}$; (Budget Maintenance)
    end if
    $\ell_t = max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$;
    if $\ell_t > 0$ **then**
      $w_{min} = \infty$;
      for $\forall i \in S_{t-1}$ **do**
        if ($f^i_{t-1} \leq 1$) **then**
          if ($y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) \leq w_{min}$) **then**
            $w_{min} = y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
            $(\mathbf{x}_b, y_b) = (\mathbf{x}_i, y_i)$;
          end if
        end if
      end for
      $f^t_{t-1} = y_t f_{t-1}(\mathbf{x}_t)$;
      $S_t = S_{t-1} \cup \{t\}$;
      if ($w_{min} \leq -\rho$) **then**
        Compute γ_t and d_{γ_b} using equation (11);
        for $\forall i \in S_t$ **do**
          $f^i_t \leftarrow f^i_{t-1} + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) + y_i d_{\gamma_b} y_b \kappa(\mathbf{x}_i, \mathbf{x}_b)$;
        end for
        $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot) + d_{\gamma_b} y_b \kappa(\mathbf{x}_b, \cdot)$;
      else /* no auxiliary example found */
        $\gamma_t = min(C, \ell_t / \kappa(\mathbf{x}_t, \mathbf{x}_t))$;
        for $\forall i \in S_t$ **do**
          $f^i_t \leftarrow f^i_{t-1} + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
        end for
        $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$;
      end if
    else
      $f_t = f_{t-1}$; $S_t = S_{t-1}$;
      for $\forall i \in S_t$ **do**
        $f^i_t \leftarrow f^i_{t-1}$;
      end for
    end if
  end if
end for
return f_T, S_T
END

Fig. 1. The Algorithms of Budget Double Updating Online Learning (BDUOL)

the support vector size of the updated f_{t-1} is smaller than B. The details of the proposed Budget DUOL (BDUOL) algorithmic framework are summarized in Algorithm 1.

The key challenge of BDUOL is to choose an appropriate reduction term Δf_{t-1} by a proper budget maintenance strategy, which can only meet the budget requirement but also minimize the impact of the reduction on the prediction performance. Unlike some existing heuristic budget maintenance approaches, in this paper, we propose a principled approach for developing several different budget maintenance strategies. Before presenting the detailed strategies, in the following, we analyze the theoretical underpinning of the proposed budget double updating online learning scheme, which is the theoretical foundation for the proposed budget maintenance strategies in Section 3.

2.4 Theoretical Analysis

In this section, we analyze the mistake bound of the proposed BDUOL algorithm. To simplify the analysis, the primal-dual framework is used to derive the mistake bound following the strategy of DUOL algorithm. Through this framework, we will show that the gap between the mistake bound of DUOL and BDUOL is bounded by the cumulative dual ascent induced by the function reduction, i.e., $\Delta f = \sum \Delta\gamma_i y_i \kappa(\mathbf{x}_i, \cdot)$. To facilitate the analysis, we firstly introduce the following lemma, which provides the dual objective function of the SVM.

Lemma 1. *The dual objective of* $\mathcal{P}_t(f) = \frac{1}{2}\|f\|_{\mathcal{H}_\kappa} + C\sum_{i=1}^{t} \ell(f(\mathbf{x}_i), y_i)$, $C > 0$ *is*

$$\mathcal{D}_t(\gamma_1, \ldots, \gamma_t) = \sum_{i=1}^{t} \gamma_i - \frac{1}{2}\|\sum_{i=1}^{t} \gamma_i y_i \kappa(\mathbf{x}_i, \cdot)\|^2_{\mathcal{H}_\kappa}, \quad \gamma_i \in [0, C], \tag{3}$$

where the relation between f *and* γ_i, $i = 1, \ldots, t$ *is* $f(\cdot) = \sum_{i=1}^{t} \gamma_i y_i \kappa(\mathbf{x}_i, \cdot)$.

According to the above lemma, after the t-th budget maintenance, the resultant dual ascent will be computed as follows:

Theorem 1. *The* ***D****ual* ***A****scent* $DA_t = \mathcal{D}_t(\gamma_1 - \Delta\gamma_1, \ldots, \gamma_t - \Delta\gamma_t) - \mathcal{D}_t(\gamma_1, \ldots, \gamma_t)$ *for the* t*-th budget maintenance, i.e.,* $f_t = f_t - \Delta f_t$*, is given as follows:*

$$DA_t = -\sum_{i=1}^{t} \Delta\gamma_i + \sum_{i=1}^{t} \Delta\gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2}\|\Delta f_t\|^2_{\mathcal{H}_\kappa}. \tag{4}$$

Proof.

$$\begin{aligned}
&\mathcal{D}_t(\gamma_1 - \Delta\gamma_1, \ldots, \gamma_t - \Delta\gamma_t) - \mathcal{D}_t(\gamma_1, \ldots, \gamma_t) \\
&= \sum_{i=1}^{t}(\gamma_i - \Delta\gamma_i) - \frac{1}{2}\|\sum_{i=1}^{t}(\gamma_i - \Delta\gamma_i) y_i \kappa(\mathbf{x}_i, \cdot)\|^2_{\mathcal{H}_\kappa} - [\sum_{i=1}^{t} \gamma_i - \frac{1}{2}\|\sum_{i=1}^{t} \gamma_i y_i \kappa(\mathbf{x}_i, \cdot)\|^2_{\mathcal{H}_\kappa}] \\
&= -\sum_{i=1}^{t} \Delta\gamma_i + \frac{1}{2}[\|f_t\|^2_{\mathcal{H}_\kappa} - \|f_t - \Delta f_t\|^2_{\mathcal{H}_\kappa}]
\end{aligned}$$

$$= -\sum_{i=1}^{t} \Delta\gamma_i + \frac{1}{2}[\|f_t\|^2_{\mathcal{H}_\kappa} - \|f_t\|^2_{\mathcal{H}_\kappa} + 2\langle f_t, \Delta f_t\rangle - \|\Delta f_t\|^2_{\mathcal{H}_\kappa}]$$

$$= -\sum_{i=1}^{t} \Delta\gamma_i + \langle f_t, \Delta f_t\rangle - \frac{1}{2}\|\Delta f_t\|^2_{\mathcal{H}_\kappa}$$

$$= -\sum_{i=1}^{t} \Delta\gamma_i + \langle f_t, \sum_{i=1}^{t} \Delta\gamma_i y_i \kappa(\mathbf{x}_i, \cdot)\rangle - \frac{1}{2}\|\Delta f_t\|^2_{\mathcal{H}_\kappa}$$

$$= -\sum_{i=1}^{t} \Delta\gamma_i + \sum_{i=1}^{t} \Delta\gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2}\|\Delta f_t\|^2_{\mathcal{H}_\kappa}.$$

Based on the above dual ascent for budget maintenance, we can now analyze the mistake bound of the proposed BDUOL algorithm. To ease our discussion, we first introduce the following lemma [4] about the mistake bound of DUOL.

Lemma 2. *Let* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ *be a sequence of examples, where* $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$ *and* $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ *for all* t, *and assume* $C \geq 1$. *Then for any function* f *in* $\mathcal{H}_\kappa$, *the number of prediction mistakes* M *made by DUOL on this sequence of examples is bounded by:*

$$2 \min_{f \in \mathcal{H}_\kappa} \left\{ \frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa} + C \sum_{i=1}^{T} \ell(f(x_i), y_i) \right\} - \frac{\rho^2}{2} M_d^w(\rho) - \frac{1+\rho}{1-\rho} M_d^s(\rho),$$

where $\rho \in [0, 1)$, $M_d^w(\rho) > 0$ *and* $M_d^s(\rho) > 0$.

Combining the above lemma with Theorem 1, it is not difficult to derive the following mistake bound for the proposed BDUOL algorithm.

Theorem 2. *Let* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ *be a sequence of examples, where* $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$ *and* $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ *for all* t, *and assume* $C \geq 1$. *Then for any function* f *in* $\mathcal{H}_\kappa$, *the number of prediction mistakes* M *made by BDUOL on this sequence of examples is bounded by:*

$$2 \min_{f \in \mathcal{H}_\kappa} \left\{ \frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa} + C \sum_{i=1}^{T} \ell(f(x_i), y_i) \right\} - 2\sum_{i=1}^{T} DA_i - \frac{\rho^2}{2} M_d^w(\rho) - \frac{1+\rho}{1-\rho} M_d^s(\rho),$$

where $\rho \in [0, 1)$.

Proof. According to the proof of Lemma 2 [4], we have

$$\frac{1}{2} M_s + \frac{1+\rho^2}{2} M_d^w(\rho) + \frac{1}{1-\rho} M_d^s(\rho) \leq \min_{f \in \mathcal{H}_\kappa} \left\{ \frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa} + C \sum_{i=1}^{T} \ell(f(x_i), y_i) \right\},$$

and $M = M_s + M_d^w(\rho) + M_d^s(\rho)$. Furthermore, by taking the dual ascents of budget maintenance into consideration, we have

$$\sum_{i=1}^{T} DA_i + \frac{M_s}{2} + \frac{1+\rho^2}{2} M_d^w(\rho) + \frac{M_d^s(\rho)}{1-\rho} \leq \min_{f \in \mathcal{H}_\kappa} \left\{ \frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa} + C \sum_{i=1}^{T} \ell(f(x_i), y_i) \right\}.$$

Rearranging the above inequality will concludes the theorem.

According to the above theorem, we can see that, if there is no budget maintenance step, the mistake bound of BDUOL is reduced to the previous mistake bound for the regular DUOL algorithm. This theorem indicates that in order to minimize the mistake bound of BDUOL, one should try to maximize the cumulative dual ascent, i.e., $\sum_{i=1}^{T} DA_i$, when designing an appropriate budget maintenance strategy.

3 Budget Maintenance Strategies

In this section, we follow the above theoretical results to develop several different budget maintenance strategies in a principled approach. In particular, as revealed by Theorem 2, a key to improving the mistake bound of BDUOL is to maximize the cumulative dual ascent $\sum_{t=1}^{T} DA_t$ caused by the budget maintenance. To achieve this purpose, we propose to maximize the dual ascent caused by budget maintenance at each online learning step, i.e.,

$$\max_{\Delta\gamma_1,\ldots,\Delta\gamma_t} DA_t = -\sum_{i=1}^{t} \Delta\gamma_i + \sum_{i=1}^{t} \Delta\gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2}\|\Delta f_t\|^2_{\mathcal{H}_\kappa}. \tag{5}$$

Below, we propose three different budget maintenance strategies, and analyze the principled approach of achieving the best dual ascent as well as the time complexity and memory cost for each strategy.

3.1 BDUOL Algorithm by Removal Strategy

The first strategy for budget maintenance is the removal strategy that discards one of existing support vectors, which is similar to the strategies used by Forgetron [8] and RBP [9]. Unlike the previous heuristic removal strategy, the key idea of our removal strategy is to discard the support vector which can maximize the dual ascent by following our previous analysis.

Specifically, let us assume the j-th SV is selected for removal. We then have the following function reduction term:

$$\Delta f_t = \gamma_j y_j \kappa(\mathbf{x}_j, \cdot). \tag{6}$$

As a result, the optimal removal solution is to discard the SV which can maximize the following dual ascent term:

$$DA_{t,j} = -\gamma_j(1 - y_j f_t(\mathbf{x}_j)) - \frac{1}{2}(\gamma_j)^2 \kappa(\mathbf{x}_j, \mathbf{x}_j). \tag{7}$$

We note that the above removal strategy is similar with the one in [5], when the Gaussian kernel is adopted where $\kappa(\mathbf{x}_j, \mathbf{x}_j) = 1$. However, our strategy is strongly theoretically motivated.

Complexity Analysis. Since the BDUOL algorithm will cache the value $y_j f_t(\mathbf{x}_j)$ for every SV, the computational complexity of $DA_{t,j}$ is $O(1)$ in practice. Thus, this

BDUOL algorithm requires $O(B)$ time complexity of computing all the $DA_{t,j}$'s. After removing the SV with the largest $DA_{t,j}$, the complexity of updating all the values of $y_j f_t(\mathbf{x}_j)$ is $O(B)$. Furthermore, combining the above discussion with the fact that the original DUOL's time complexity is $O(B)$, we can conclude that the overall time complexity of this BDUOL algorithm is also $O(B)$. As for the memory cost, since only B SVs, their weight parameters and the $y_j f_t(\mathbf{x}_j)$s have to be cached, the space complexity is thus also $O(B)$.

3.2 BDUOL Algorithm by Projection Strategy

Although the above removal strategy is optimized to find the best support vector that maximizes the dual ascent (i.e., minimizes the loss of dual ascent caused by budget maintenance), it is still unavoidable to result in the loss of the dual ascent due to the removal of one existing support vector. To minimize such loss, we propose a projection strategy for budget maintenance.

Specifically, in the projection strategy, the selected j-th SV for removal will be projected to the space spanned by the rest SVs. The objective is to find the function closest to $\gamma_j y_j \kappa(\mathbf{x}_j, \cdot)$ in the space spanned by the remaining SVs, or formally:

$$\min_{\beta_i \in [-\gamma_i, C-\gamma_i], i \neq j} \|\gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \sum_{i \neq j} \beta_i y_i \kappa(\mathbf{x}_i, \cdot)\|^2_{\mathcal{H}_\kappa}. \tag{8}$$

which is essentially a Quadratic Programming (QP) problem. So, we can exploit the existing efficient QP solvers to find the optimal solution.

However, solving the above QP problem directly may not be efficient enough for online learning purpose. To further improve the efficiency, we also proposed an approximate solution by firstly solving the unconstrained optimization problem and then projecting the solution into the feasible region of the constraints. Specifically, setting the gradient of the above equation with respect to $\overline{\beta} = [\beta_i]^\top$, $i \neq j$ as zero, one can obtain the optimal solution as

$$\overline{\beta} = \gamma_j y_j \mathbf{K}^{-1} \mathbf{k}_j ./ \overline{y}, \tag{9}$$

where $\mathbf{K}$ is the kernel matrix for $\mathbf{x}_i, i \neq j$, $\mathbf{k}_j = [\kappa(\mathbf{x}_i, \mathbf{x}_j)]^\top$, $i \neq j$, $./$ is element-wise division and $\overline{y} = [y_i]^\top$, $i \neq j$. In the above, inverting $\mathbf{K}$ can be efficiently realized by using Woodbury formula [12]. As a result, we should set

$$\Delta f_t(\cdot) = \gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \sum_{i \neq j} \beta_i y_i \kappa(\mathbf{x}_i, \cdot). \tag{10}$$

However, the resultant $f_t - \Delta f_t$'s SV weights may not fall into the range $[0, C]$. To fix this problem, we will project each β_i as follows:

$$\overline{\beta} = \Pi_{[-\overline{\gamma}, C-\overline{\gamma}]}(\gamma_j y_j \mathbf{K}^{-1} \mathbf{k}_j ./ \overline{y}), \tag{11}$$

where $\Pi_{[a,b]}(x) = \max(a, \min(b, x))$ and $\overline{\gamma} = [\gamma_i]^\top$, $i \neq j$.

Now the key problem is to find the best SV among $B+1$ candidates for projection. Since after the projection of $\gamma_j y_j \kappa(\mathbf{x}_j, \cdot)$, the resultant dual ascent is

$$DA_{t,j} = -\sum_i \Delta\gamma_i + \sum_i \Delta\gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2}\|\Delta f_t\|^2_{\mathcal{H}_\kappa}. \tag{12}$$

So the best SV is the one which can achieve the largest value $DA_{t,j}$.

Complexity Analysis. The time complexity for BDUOL is dominated by the computation K^{-1}. Computing K^{-1} using K will cost $O(B^3)$ time, however using Woodbury formula, we can efficiently compute it using only $O(B^2)$ time. In addition, we need to compute B times projections for every SV, so the total time complexity for one step of updating is $O(B^3)$. Finally, the memory burden for the BDUOL algorithm is $O(B^2)$, since the storage of kernel matrix K and inverse kernel matrix K^{-1} dominated the main memory cost.

3.3 BDUOL Algorithm by Nearest Neighbor Strategy

The above projection strategy is able to achieve a better improvement of dual ascent than the removal strategy, it is however much more computationally expensive. To balance the tradeoff between efficiency and effectiveness, we propose an efficient nearest neighbor strategy which approximates the projection strategy by projecting the removed SV to its nearest neighbor SV, based on the distance in the mapped feature space. This strategy is motivated by the fact that the nearest neighbor SV usually could be a good representative of the removed SV. Using this strategy, we can significantly improve the time efficiency. In particular, as we use only the nearest neighbor for projection, the corresponding solution according to equation 11 can be expressed:

$$\beta_{N_j} = \Pi_{[-\gamma_{N_j}, C-\gamma_{N_j}]}(\gamma_j y_j \kappa(\mathbf{x}_{N_j}, \mathbf{x}_{N_j})^{-1} \kappa(\mathbf{x}_{N_j}, \mathbf{x}_j)/y_{N_j}), \tag{13}$$

where $\kappa(\mathbf{x}_{N_j}, \cdot)$ is the nearest neighbor of $\kappa(\mathbf{x}_j, \cdot)$. As a result, the corresponding $\Delta f_t = \gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \beta_{N_j} y_{N_j} \kappa(\mathbf{x}_{N_j}, \cdot)$. Since after the projection of $\gamma_j y_j \kappa(\mathbf{x}_j, \cdot)$, the resultant dual ascent is

$$DA_{t,j} = -\sum_i \Delta\gamma_i + \sum_i \Delta\gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2}\|\Delta f_t\|^2_{\mathcal{H}_\kappa}. \tag{14}$$

Thus, the best SV is the one that has the largest value $DA_{t,j}$.

Complexity Analysis. All the computation steps except looking for the nearest neighbor have constant time complexity of $O(1)$. For improving nearest neighbor searching, we can cache the indexes and the distances about the nearest neighbors of the SVs, making finding the nearest neighbor in $O(1)$. After remove the best SV, we should also update the caches. The time complexity for this updating is at most $O(B)$. In summary, the time complexity for the overall strategy is $O(B)$, and the overall memory cost is also $O(B)$.

Remark: To improve the efficiencies of the projection and nearest neighbor strategies, we could use the projection or the nearest neighbor strategy to keep the information from the SV selected by the removal strategy and then remove it, since the search cost of the removal strategy is quite lower than the other two strategies.

4 Experimental Results

In this section, we evaluate the empirical performance of the proposed algorithms for Budget Double Updating Online Learning (BDUOL) by comparing them with the state-of-the-art algorithms for budget online learning.

4.1 Algorithms for Comparison

In our experiments, we implement the proposed BDUOL algorithms as follows:

- "BDUOL$_{remo}$": the BDUOL algorithm by the **remo**val strategy for budget maintenance described in section 3.1,
- "BDUOL$_{proj}$": the BDUOL algorithm by the exact **proj**ection strategy by a standard QP solver for budget maintenance described in section 3.2,
- "BDUOL$_{appr}$": the BDUOL algorithm by the **appr**oximate projection strategy for budget maintenance described in section 3.2,
- "BDUOL$_{near}$": the BDUOL algorithm by the **near**est neighbor strategy for budget maintenance described in section 3.3,

For comparison, we include the following state-of-the-art algorithms for budget online learning:

- "RBP": the Random Budget Perceptron algorithm [9],
- "Forgetron": the Forgetron algorithm [8],
- "Projectron": the Projectron algorithm [10], and
- "Projectron++": the aggressive version of Projectron algorithm [10].

Besides, we also include two non-budget online learning algorithms as yardstick:

- "Perceptron": the classical Perceptron algorithm [6], and
- "DUOL": the Double Updating Online Learning algorithm [4].

4.2 Experimental Testbed and Setup

We test all the algorithms on six benchmark data sets from web machine learning repositories listed in Table 1. These data sets can be downloaded from LIBSVM website[1], UCI machine learning repository[2], and MIT CBCL face data sets[3]. These datasets were chosen fairly randomly to cover various sizes of datasets.

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[2] http://www.ics.uci.edu/~mlearn/MLRepository.html
[3] http://cbcl.mit.edu/software-datasets

Table 1. Details of the datasets in our experiments

Dataset	# instances	# features
german	1000	24
MITface	6977	361
mushrooms	8124	112
spambase	4601	57
splice	3175	60
w7a	24692	300

To make a fair comparison, all the algorithms in our comparison adopt the same experimental setup. A gaussian kernel is adopted in our study, for which the kernel width is set to 8 for all the algorithms and datasets. To make the number of support vectors fixed for Projectron and Projectron++ algorithms, we simply store the received SVs before the budget overflows, and then project the subsequent ones into the space spanned by the stored SVs afterward. The penalty parameter C in the DUOL algorithm was selected by 5-fold cross validation for all the datasets from range $2^{[-10:10]}$. Due to the cross-validation, we randomly divide every dataset into two equal subsets: cross validation (CV) dataset and online learning dataset. And C is set as the same value with DUOL. Furthermore, the value ρ for the DUOL and its budget variants is set as 0, according to the previous study on the effect of ρ.

The budget sizes B for different datasets are set as proper fractions of the support vector size of Perceptron, which are shown in Table 3. All the experiments were conducted 20 times, each with a different random permutation of data points. All the results were reported by averaging over the 20 runs. For performance metrics, we evaluate the online classification performance by evaluating online cumulative mistake rates and running time cost.

4.3 Performance Evaluation of Non-budget Algorithms

Table 2 summarizes the average performance of the two non-budget algorithms for kernel-based online learning. First of all, similar to the previous study [4], we found that DUOL outperforms Perceptron significantly for all the datasets

Table 2. Evaluation of non-budget algorithms on the the data sets

Algorithm	Perceptron			DUOL		
Datasets	Mistake (%)	Support Vectors (#)	Time (s)	Mistakes (%)	Support Vectors (#)	Time (s)
german	35.760 %± 1.149	178.800 ± 5.745	0.007	29.820 %± 1.243	381.750 ± 5.866	0.044
MITface	6.246 %± 0.252	217.85 ± 8.774	0.04	2.418 %± 0.156	408.5 ± 9.339	0.114
mushrooms	3.175 %± 0.463	128.950 ± 18.805	0.040	0.591 %± 0.086	247.050 ± 14.084	0.099
spambase	27.354 %± 0.561	629.150 ± 12.906	0.050	22.680 %± 0.557	1385.800 ± 17.519	0.317
splice	21.808 %± 0.709	346.100 ± 11.257	0.026	15.633 %± 0.461	777.450 ± 11.551	0.130
w7a	4.366 %± 0.093	539.000 ± 11.530	0.272	3.068 %± 0.114	1171.600 ± 30.396	0.728

Table 3. Evaluation of several budgeted algorithms with varied budget sizes

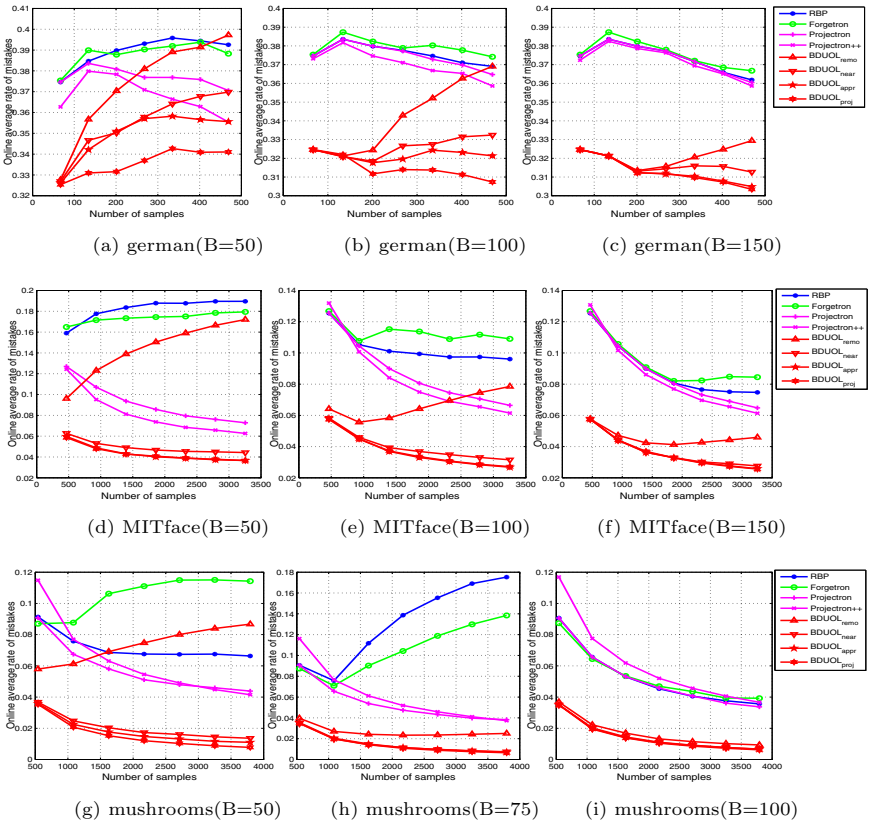
Budget Size		B=50		B=100		B=150	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
german	RBP	39.140 %± 1.338	0.010	36.940 %± 1.837	0.009	36.080 %± 1.392	0.008
	Fogetron	38.840 %± 1.551	0.014	37.250 %± 1.399	0.013	36.570 %± 1.641	0.013
	Projectron	36.990 %± 1.652	0.020	36.310 %± 1.441	0.027	35.870 %± 1.149	0.063
	Projectron++	35.370 %± 1.413	0.036	35.620 %± 1.251	0.046	35.680 %± 1.380	0.103
	BDUOL_{remo}	39.970 %± 3.150	0.068	37.180 %± 2.297	0.077	33.330 %± 2.264	0.088
	BDUOL_{near}	37.090 %± 1.763	0.098	33.330 %± 1.697	0.112	31.360 %± 1.511	0.129
	BDUOL_{appr}	35.540 %± 2.010	0.160	32.340 %± 1.570	0.381	30.450 %± 1.338	0.941
	BDUOL_{proj}	**34.030** %± 1.104	0.214	**30.710** %± 1.261	0.624	**30.330** %± 1.221	1.341
Budget Size		B=50		B=100		B=150	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
MITface	RBP	18.964 %± 1.330	0.063	9.557 %± 0.615	0.053	7.463 %± 0.609	0.050
	Fogetron	18.038 %± 1.470	0.081	10.707 %± 0.883	0.077	8.373 %± 0.546	0.076
	Projectron	7.137 %± 0.384	0.086	6.461 %± 0.288	0.098	6.316 %± 0.304	0.142
	Projectron++	6.135 %± 0.312	0.149	5.973 %± 0.215	0.197	5.978 %± 0.219	0.373
	BDUOL_{remo}	17.516 %± 2.264	0.225	8.096 %± 0.811	0.208	4.700 %± 0.449	0.194
	BDUOL_{near}	4.435 %± 0.396	0.179	3.078 %± 0.304	0.187	2.701 %± 0.277	0.205
	BDUOL_{appr}	3.632 %± 0.277	0.249	**2.618** %± 0.221	0.429	2.501 %± 0.258	0.836
	BDUOL_{proj}	**3.611** %± 0.319	0.273	2.645 %± 0.228	0.492	**2.481** %± 0.168	0.755
Budget Size		B=50		B=75		B=100	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
mushrooms	RBP	6.551 %± 1.056	0.054	17.841 %± 1.539	0.073	3.488 %± 0.713	0.049
	Fogetron	11.273 %± 1.750	0.085	14.154 %± 2.748	0.094	3.895 %± 1.141	0.079
	Projectron	4.264 %± 0.613	0.095	3.703 %± 0.731	0.099	3.207 %± 0.473	0.107
	Projectron++	3.986 %± 0.297	0.161	3.557 %± 0.174	0.178	3.484 %± 0.117	0.197
	BDUOL_{remo}	8.754 %± 2.438	0.210	2.547 %± 0.865	0.181	0.891 %± 0.152	0.162
	BDUOL_{near}	1.329 %± 0.198	0.141	0.710 %± 0.095	0.132	0.618 %± 0.081	0.131
	BDUOL_{appr}	1.065 %± 0.218	0.193	0.667 %± 0.092	0.194	0.623 %± 0.080	0.248
	BDUOL_{proj}	**0.729** %± 0.060	0.230	**0.604** %± 0.080	0.266	**0.577** %± 0.071	0.290
Budget Size		B=200		B=400		B=600	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
spambase	RBP	31.826 %± 0.924	0.065	29.220 %± 0.550	0.069	27.417 %± 0.598	0.059
	Fogetron	32.461 %± 0.971	0.077	29.641 %± 0.742	0.084	27.424 %± 0.644	0.082
	Projectron	29.237 %± 0.750	0.238	27.480 %± 0.484	0.929	27.750 %± 2.134	2.776
	Projectron++	28.822 %± 0.725	0.819	28.693 %± 6.781	3.874	27.559 %± 1.572	7.277
	BDUOL_{remo}	34.559 %± 1.308	0.522	28.989 %± 0.927	2.013	25.661 %± 0.709	4.541
	BDUOL_{near}	28.180 %± 1.084	0.756	25.607 %± 0.748	3.874	24.187 %± 0.584	8.570
	BDUOL_{appr}	28.950 %± 2.001	5.555	26.843 %± 0.948	16.396	24.846 %± 0.695	30.186
	BDUOL_{proj}	**27.448** %± 0.961	8.837	**25.307** %± 0.793	53.987	**23.780** %± 0.640	197.199
Budget Size		B=100		B=200		B=300	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
splice	RBP	30.003 %± 1.318	0.035	25.126 %± 0.900	0.033	22.530 %± 0.967	0.028
	Fogetron	29.912 %± 1.483	0.044	25.545 %± 0.582	0.043	22.457 %± 0.734	0.041
	Projectron	22.851 %± 0.756	0.061	22.007 %± 1.023	0.146	21.830 %± 0.811	0.356
	Projectron++	22.628 %± 0.649	0.163	21.843 %± 1.002	0.445	21.919 %± 1.188	0.922
	BDUOL_{remo}	26.150 %± 1.279	0.238	22.117 %± 1.321	0.409	18.299 %± 0.797	0.641
	BDUOL_{near}	23.125 %± 1.161	0.435	18.847 %± 0.866	0.458	16.991 %± 0.600	1.201
	BDUOL_{appr}	**19.779** %± 0.754	1.110	18.062 %± 2.000	2.596	17.927 %± 2.065	3.905
	BDUOL_{proj}	20.243 %± 0.667	1.041	**17.190** %± 0.734	2.685	**16.191** %± 0.517	5.475
Budget Size		B=300		B=400		B=500	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
w7a	RBP	4.681 %± 0.257	0.291	4.597 %± 0.227	0.316	4.434 %± 0.108	0.332
	Fogetron	4.697 %± 0.102	0.367	4.584 %± 0.180	0.390	4.466 %± 0.137	0.398
	Projectron	4.680 %± 0.290	0.727	4.625 %± 0.456	1.174	4.406 %± 0.130	1.929
	Projectron++	3.880 %± 0.526	4.366	3.672 %± 0.214	7.763	3.666 %± 0.152	8.796
	BDUOL_{remo}	3.832 %± 0.170	0.995	3.361 %± 0.145	1.824	**3.175** %± 0.133	2.401
	BDUOL_{near}	3.572 %± 0.114	1.241	3.269 %± 0.133	3.324	3.183 %± 0.087	4.859
	BDUOL_{appr}	4.126 %± 0.502	4.553	4.001 %± 0.153	6.794	3.775 %± 0.121	9.646
	BDUOL_{proj}	**3.367** %± 0.096	11.306	**3.227** %± 0.126	18.965	3.254 %± 0.332	29.807

according to t-test results, which validates our motivation of choosing DUOL as the basic online learning algorithm for budget online learning. Second, we noticed that the support vector size of DUOL is in general much larger than that of Perceptron. Finally, the time cost of DUOL is much higher than that of Perceptron, mostly due to the larger number of support vectors. Both the large number of support vectors and high computational time motivate the need of studying budget DUOL algorithms in this work.

4.4 Performance Evaluation of Budget Algorithms

Table 3 summarizes the results of different budget online learning algorithms. We can draw several observations.

First of all, we observe that RBP and Forgetron achieve very similar performance for most cases. In addition, we also find that Projectron++ achieves a lower mistake rate than Projectron for almost all the datasets and for varied

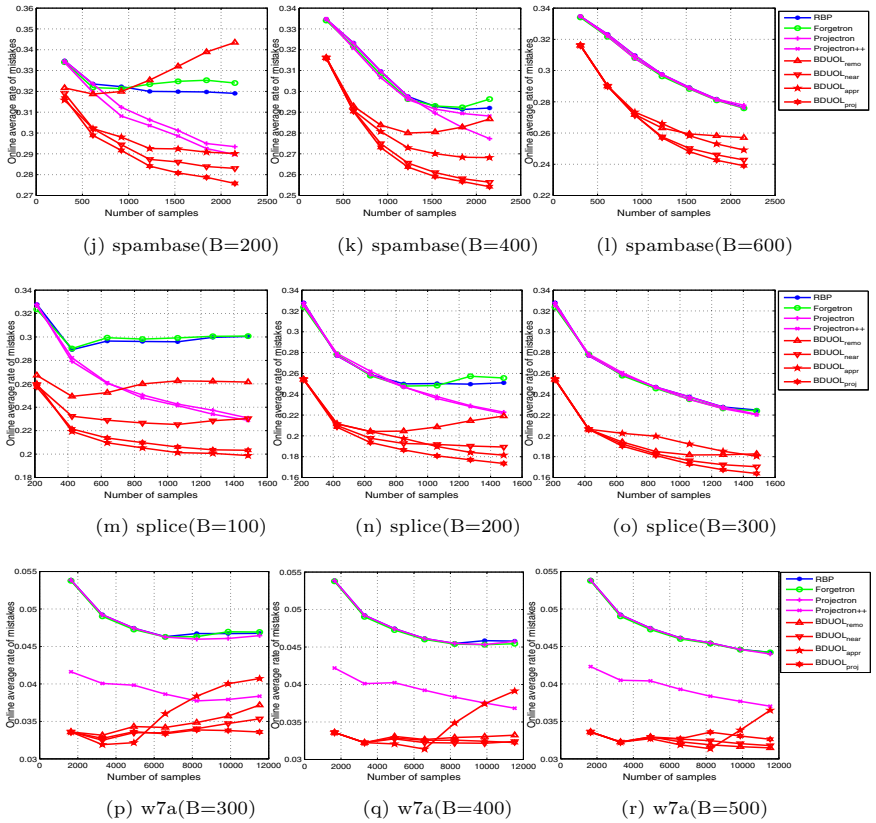


(a) german(B=50) (b) german(B=100) (c) german(B=150)

(d) MITface(B=50) (e) MITface(B=100) (f) MITface(B=150)

(g) mushrooms(B=50) (h) mushrooms(B=75) (i) mushrooms(B=100)

Fig. 2. Evaluation of online mistake rates against the number of samples on three datasets. The plotted curves are averaged over 20 random permutations.



(j) spambase(B=200) (k) spambase(B=400) (l) spambase(B=600)

(m) splice(B=100) (n) splice(B=200) (o) splice(B=300)

(p) w7a(B=300) (q) w7a(B=400) (r) w7a(B=500)

Fig. 3. Evaluation of online mistake rates against the number of samples on three datasets. The plotted curves are averaged over 20 random permutations.

budget sizes, which is similar to the previous results reported in [10]. Moreover, compared with the baseline algorithms RBP and Forgetron, the proposed BDUOL$_{remo}$ algorithm using a simple removal strategy achieves comparable or better mistake rate when the budget size is large, but fails to improve when the budget size is very small, which indicates a simple removal strategy may not be always effective and a better budget maintenance strategy is needed.

Second, among all the algorithms in comparison for budget online learning, we find that BDUOL$_{proj}$ always achieves the lowest mistake rates for most cases. These promising results indicate the projection strategy can effectively reduce the information loss. However, we also notice that the time cost of the BDUOL$_{proj}$ is among the highest ones, which indicates it is important to find some more efficient strategy.

Third, by comparing two approximate strategies, we find that BDUOL$_{appr}$ achieves better mistake rates than BDUOL$_{near}$ only on the german and mushrooms datasets, while it consumes too much time than the proposed BDUOL$_{near}$ algorithms, which indicates BDUOL$_{near}$ achieves better trade off between

mistake rates and time complexity than BDUOL$_{appr}$. In addition, when the number of budget is large, BDUOL$_{near}$ always achieves similar performance with the BDUOL$_{proj}$, while consumes significantly less time, which indicates the proposed nearest neighbor strategy is a good alternative of the projection strategy.

Finally, Figure 2 and Figure 3 show the detailed online evaluation processes of the several budget online learning algorithms. Similar observations from these figures further verified the efficacy of the proposed BDUOL technique.

5 Conclusions

This paper presented a new framework of budget double updating online learning for kernel-based online learning on a fixed budget, which requires the number of support vectors associated with the prediction function is always bounded by a predefined budget. We theoretically analyzed its performance, which reveals that its effectiveness is tightly connected with the dual ascent achieved by the model reduction for budget maintenance. Based on the theoretical analysis, we proposed three budget maintenance strategies: removal, projection, and nearest neighbor. We evaluate the proposed algorithms extensively on benchmark datasets. The promising empirical results show that the proposed algorithms outperform the state-of-the-art budget online learning algorithms in terms of mistake rates. Future work will exploit different budget maintenance strategies and extend the proposed work to multi-class budgeted online learning.

Acknowledgments. This work was supported by Singapore MOE tier 1 grant (RG33/11) and Microsoft Research grant (M4060936).

References

1. Kivinen, J., Smola, A.J., Williamson, R.C.: Online learning with kernels. IEEE Transactions on Signal Processing 52(8), 2165–2176 (2004)
2. Cheng, L., Vishwanathan, S.V.N., Schuurmans, D., Wang, S., Caelli, T.: Implicit online learning with kernels. In: NIPS, pp. 249–256 (2006)
3. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research 7, 551–585 (2006)
4. Zhao, P., Hoi, S.C.H., Jin, R.: Double updating online learning. Journal of Machine Learning Research 12, 1587–1615 (2011)
5. Crammer, K., Kandola, J.S., Singer, Y.: Online classification on a budget. In: NIPS (2003)
6. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65, 386–407 (1958)
7. Weston, J., Bordes, A.: Online (and offline) on an even tighter budget. In: AISTATS, pp. 413–420 (2005)
8. Dekel, O., Shalev-Shwartz, S., Singer, Y.: The forgetron: A kernel-based perceptron on a budget. SIAM J. Comput. 37(5), 1342–1372 (2008)

9. Cavallanti, G., Cesa-Bianchi, N., Gentile, C.: Tracking the best hyperplane with a simple budget perceptron. Machine Learning 69(2-3), 143–167 (2007)
10. Orabona, F., Keshet, J., Caputo, B.: Bounded kernel-based online learning. Journal of Machine Learning Research 10, 2643–2666 (2009)
11. Vapnik, V.N.: Statistical Learning Theory. Wiley (1998)
12. Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. In: Advances in Neural Information Processing Systems 13 (NIPS), pp. 409–415 (2000)

Handling Time Changing Data with Adaptive Very Fast Decision Rules

Petr Kosina[1] and João Gama[2]

[1] LIAAD-INESC Porto, FI Masaryk University, Czech Republic
petr.kosina@inescporto.pt
[2] LIAAD-INESC Porto, FEP-University of Porto
jgama@fep.up.pt

Abstract. Data streams are usually characterized by changes in the underlying distribution generating data. Therefore algorithms designed to work with data streams should be able to detect changes and quickly adapt the decision model. Rules are one of the most interpretable and flexible models for data mining prediction tasks. In this paper we present the Adaptive Very Fast Decision Rules (AVFDR), an on-line, any-time and one-pass algorithm for learning decision rules in the context of time changing data. AVFDR can learn ordered and unordered rule sets. It is able to adapt the decision model via incremental induction and specialization of rules. Detecting local drifts takes advantage of the modularity of rule sets. In AVFDR, each individual rule monitors the evolution of performance metrics to detect concept drift. AVFDR prunes rules that detect drift. This explicit change detection mechanism provides useful information about the dynamics of the process generating data, faster adaption to changes and generates compact rule sets. The experimental evaluation shows this method is able to learn fast and compact rule sets from evolving streams in comparison to alternative methods.

1 Motivation

Machine learning in recent years has undergone a change in the focus and application of its techniques. More and more attention is shifted towards ubiquitous mining and real-time applications as more and more data is represented by possibly infinite streams. New approaches and stream extensions to off-line methods emerge in order to improve performance under constrains given by the stream environment. It includes characteristics such as being able to produce a model while scanning the data only once, the model must be available at any point of time, must be up-to-date, and all must be able to run under computational and memory constrains [19].

One of the common phenomena in stream mining is a change in data that might occur over time. This phenomenon addresses mostly the up-to-date requirement of the stream approaches. Almost everything in this world changes and so can the concepts in data if they are being observed for long enough time. The influence of seasonal change, economical change, health conditions or wear of machinery can cause decrease of quality of the models built on previous observations. Therefore, fast adaptation of models has an advantage for most of the real world problems.

Among the best known and most used models for data stream classification are algorithms based on Hoeffding Trees [12] (HT). The adaptation in HT algorithms is present

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 827–842, 2012.

in an inexplicit form via incremental growth of the tree. The adaptation in this case is slow. Faster adaptation might be achieved by employing explicit drift detection methods, e.g. [20,1,22]. This approach, however, require rebuilding the current tree which might be inefficient. Decision rules are a classification model similar to decision trees, which has an advantage of having individual rules that can be managed independently. Therefore, in decision rules the implicit adaptation feature that is present in the trees remains but in addition to it, the set of rules can be altered more easily. Instead of rebuilding the classifier from scratch or executing a complicated change of the structure of a tree, individual rules which are considered outdated can be simply removed. This paper presents Adaptive Very Fast Decision Rules (AVFDR) classifier as an extension to VFDR [16]. The main contribution is that this system focuses on time changing data and it incorporates a drift detection mechanism for each individual rule. It is capable of faster adaptation to a change and the mechanism also serves as rule set pruning.

The paper is organized as follows. The Section 2 discusses the related work in rule learning and handling time changing data. The AVFDR system is presented in Section 3 and the experimental results are in Section 4. Finally, Section 5 concludes the paper and mentions future work directions.

2 Related Work

Decision rules are well known classification method in off-line learning with a strong connection to decision tree learning. There are approaches that build decision lists from the tree [28], where each rule then corresponds to the path from the root to a leaf and there are as many rules as leaves. The set can be then optimized to obtain simpler and more accurate classifier.

There exist many algorithms for building decision lists [29,7,9,11,34]. CN2, presented in [7], evaluates every possible conjunction of attribute tests (*if-conditions*) of a rule based on information-theoretic entropy measure. Later in [8], the entropy measure is replaced by Laplace accuracy estimate. The algorithm searches for new rules for each class in turn each time separating one (positive) from all the others (negative) as two-class problem. Only positive examples that satisfy learned rule are removed from the training set for next iteration of the rule search. RIPPER [9] orders classes in increasing order of their frequency in the training set. After learning rules that separate minority class, the covered examples are removed and the algorithm proceeds with the next class. The process stops when the last single class remains that represents the default class.

Incremental rule learners include STAGGER [30], the first system designed expressly for coping with concept drift, the FLORA family of algorithms [35] with FLORA3 being the first system able to deal with recurring contexts, and the AQ-PM family [27]. The first rule learning system designed for streaming numerical data is system Facil, presented in [13]. Facil learns decision rules that might overlap. Expansion of a rule is controlled by border examples that are stored together with rules when they are learned. Adaptation to drift is blind, by deleting older rules.

The original VFDR system, which incrementally learns new rules and specialises existing ones, was presented by [16]. It is capable of learning ordered or unordered sets and employs Bayesian leaves. The system is further extended in [26] to focus the

attention on learning rules for various classes of labeled data and it can induce multiple rules (one for each such class) at one point when specialization is evaluated.

The stream mining community has already introduced many different approaches to deal with the phenomenon of concept drift. Sliding windows and example weights [25] are widely used approaches to maintain a classifier consistent to the most recent data. In [4] the authors proposed the ADWIN algorithm, a detector and estimator which automatically adapts to the current rate of change by keeping the window of recent examples of variable length. It employs the Hoeffding bound to guarantee that the window has maximal length without a change inside the window. Other methods can explicitly detect change-points or small time-windows where the concept to learn has changed. A classifier can be equipped with such drift detection methods associated with a forgetting mechanism. Drift detection methods might monitor the evolution of the error rate, like the SPC algorithm [20], monitor the distance between classification errors, like in [1], etc.

In [23], the authors presented system CVFDT, a decision tree learner for mining data streams with non-stationary distribution. CVFDT learns model consistent with a sliding window of recent examples. When concept is changing and split that was previously executed would no longer be the best, it starts learning an alternate subtree with new best attribute as its root. The subtree replaces the original one when it becomes more accurate.

As pointed out in [33], a drawback of decision trees is that even a slight drift of the target function may trigger several changes in the model and severely compromise learning efficiency. On the other hand, ensemble methods avoid expensive revisions by weighting the members, but may run the risk of building unnecessary learners when virtual drifts are present in data. In [5] two new decision tree ensemble methods were presented: ADWIN Bagging and Adaptive-Size Hoeffding Tree bagging. The former extends on-line bagging with ADWIN change detector, which works as an estimator for the weights of the boosting method. The worst performing classifier is removed from the ensemble when change is detected and it is replaced by a new one. The latter uses Hoeffding trees of different maximum sizes since smaller trees adapt faster to changes and larger work better for long periods with little or no change.

3 Adaptive Very Fast Decision Rules

In this section we present AVFDR algorithm which extends VFDR classifier for data streams by integrating drift detection to each rule.

3.1 Growing a Set of Rules

The AVFDR algorithm is designed for high-speed data streams which learns unordered set rules and needs only one scan of data.

The algorithm begins with a empty rule set (RS) and a default rule $\{\} \rightarrow \mathcal{L}$, where $\mathcal{L}$ is initialized to $\emptyset$. $\mathcal{L}$ is a data structure that contains information used to classify test instances, and the sufficient statistics needed to expand the rule. Each learned rule (r) is a conjunction of literals, that are conditions based on attribute values, and a $\mathcal{L}_r$.

Algorithm 1: AVFDR: Rule Learning Algorithm.

```
input : S: Stream of examples
        ordered_set: boolean flag
output: RS: Set of Decision Rules
begin
    Let RS ← {}
    Let default rule ℒ ← ∅
    foreach example (x, y_k) ∈ S do
        foreach Rule r ∈ RS do
            if r covers the example then
                /* Estimate Rule Status                    */
                Status ← SPC(r, x_t, y_t)
                if Status == OutControl then
                    RS ← RS − {r}
                else
                    if Status == InControl then
                        Update sufficient statistics of r
                        RS ← RS − {r}
                        RS ← RS ∪ ExpandRule(r)
                        if ordered_set then
                            BREAK
        if none of the rules in RS covered example then
            Update sufficient statistics of the default rule
            RS ← RS ∪ ExpandEmptyRule(default rule)
```

For numerical attributes, each literal is of the form $X_i > v$, or $X_i \leq v$ for some feature X_i and some constant v. For categorical attributes AVFDR produce literals of the form $X_i = v_j$ where v_j is a value in the domain of X_i. Please note that for simplicity we use the $X_i = v_j$ notation for both cases, numerical and categorical, in the context of adding a new condition.

If all the literals are true for a given example, then the example is said to be *covered* by the rule. The labeled examples covered by a rule r are used to update $\mathcal{L}_r$. A rule is expanded with the literal that has the highest gain measure of the examples covered by the rule. $\mathcal{L}_r$ accumulates the sufficient statistics, is similar to statistics in [15], to compute the gain measure of all possible literals. $\mathcal{L}_r$ is a data structure that contains: an integer that stores the number of examples covered by the rule; a vector to compute $p(c_k)$, i.e., the probability of observing examples of class c_k; a matrix $p(X_i = v_j|c_k)$ to compute the probability of observing value v_j of a nominal attribute X_i per class; and a btree to compute the probability of observing values greater than v_j of continuous attribute X_i, $p(X_i > v_j|c_k)$, per class.

The number of observations after which a rule can be expanded or new rule can be induced is determined by Hoeffding bound. It guarantees that with the probability $1 - \delta$ the true mean of a random variable x with a range R will not differ from the

Algorithm 2: The SPC Algorithm

Input : r: rule ;
/* n^{th} observation of rule r */
Current example: $\boldsymbol{x}_n, y_n$;
Output: Status $\in$ {InControl, OutControl, Warning}
begin
  Let $\hat{y}_n \leftarrow r(\boldsymbol{x}_n)$;
  Let $error_n \leftarrow L(\hat{y}_n, y_n)$;
  Compute error's mean p_n and variance s_n;
  if $p_n + s_n < p_{min} + s_{min}$ **then**
    $p_{min} \leftarrow p_n$;
    $s_{min} \leftarrow s_n$;
  if $p_n + s_n < p_{min} + \beta \times s_{min}$ **then**
    Status $\leftarrow$ 'InControl' ;
  else
    if $p_n + s_n < p_{min} + \alpha \times s_{min}$ **then**
      Status $\leftarrow$ 'Warning';
    else
      Status $\leftarrow$ 'OutControl';
  Return: Status;

estimated mean after N independent observations by more than: $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$. It is not efficient to check for the sufficient number of examples with every incoming example, therefore it is done after only N_{min} observations.

The set of rules (RS) is learned in parallel as described in Algorithm 1. We consider two cases: learning ordered or unordered set of rules (AVFDR^o and AVFDR^u). In the former, every labeled example updates statistics of the first rule that covers it. In the latter, every labeled example updates statistics of all the rules that cover it. If a labeled example is not covered by any rule, the default rule is updated.

We can further apply different strategies to functions that are responsible for creating new rules from *default rule* and expansion of other rules.

3.2 Expansion of a rule

The AVFDR classifier applies one vs. all strategy in which examples of class $c_k \in C$ are positive and $\forall c_l \in C, c_l \neq c_k$ are negative. It considers a rule expansion for each class $c \in C_r$, where C_r is the set of classes observed at rule r. The process to select new condition for a rule works as follows.

For each attribute X_i the value of gain function G adopted from FOIL is computed. The change in gain between rule r and a candidate rule after adding a new condition r' is defined as $Gain(r', r) = s \times \left(\log_2 \frac{N'_+}{N'} - \log_2 \frac{N_+}{N} \right)$, where N is the number of examples covered by r and N_+ is the number of positive examples in them, N'_+ and

Algorithm 3: `ExpandRule`: Expanding Unordered Rule.

input : r: One Rule;
δ: Confidence
τ: Constant to solve ties
output: NR: New Rules Set;
begin
  Let $NR \leftarrow \{r\}$
  Compute $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$ (Hoeffding bound)
  `/* Expand rule for the original class */`
  Let c_r be class of r
  $EvaluateLiterals(c_r)$
  if $g^k_{best} - g^k_{2best} > \epsilon$ *or* $\epsilon < \tau$ **then**
    Extend r with a new condition based on the best attribute $X_a = v_j$
    `/* Expand rule for other classes */`
    foreach *class* $c_k \neq c_r$ **do**
      $EvaluateLiterals(c_k)$
      if $g^k_{best} - g^k_{2best} > \epsilon$ *or* $\epsilon < \tau$ **then**
        create new r' by extending r with a new condition $X_a = v_j$ and class c_k
        $NR \leftarrow NR \cup \{r'\}$
    Release sufficient statistics of r
  return NR

N' represent the same for r', and s is the number of true positives in r that are still true positives in r', which in this case corresponds to N'_+.

We are interested only in positive gain, therefore we consider the minimum of the gain function as 0 and the maximum for a given rule is $N_+ \times \left(-\log_2 \frac{N_+}{N}\right)$. We can then normalize the positive gain as: $GainNorm(r', r) = \frac{Gain(r', r)}{N_+ \times \left(-\log_2 \frac{N_+}{N}\right)}$.

G is computed for each attribute value v_j, which was observed in more than certain fraction of examples (e.g., 10% of examples) and class c_k. Procedure in Algorithm 6 searches for the best and second best value of $G()$ for given class. If g^k_{best} is the true best gain measure, i.e., satisfies condition $g^k_{best} - g^k_{2best} > \epsilon$ for given class value c_k, the rule is expanded with condition $X_a = v_j \Rightarrow c_k$.

Algorithm 5 describes the expansion of *default rule* to a new rule of the rule set. The new literal of the new rule has the best attribute-value evaluation and its positive class is the one with the minimum frequency among those that satisfy the Hoeffding bound condition. The search for new rules continues until the expansions for all possible classes are checked.

In case of expanding some rule that already contains conditions the procedure works as follows. The rule was induced for a certain class that was set as positive. To keep this class of interest in the set, it is maintained as positive for the next computations of the measure criterion (Algorithm 4). The unordered rule set in Algorithm 3 considers computations of other classes set as the positive one and expanding the rule with the

Algorithm 4: `ExpandRule`: Expanding Ordered Rule.

input : r: One Rule;
δ: Confidence
τ: Constant to solve ties
output: NR: New Rules Set;
begin
  Let $NR \leftarrow \{r\}$
  Let c be the class of rule r
  Compute $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$ (Hoeffding bound)
  Let c_r be class of r
  $EvaluateLiterals(c_r)$
  if $g^k_{2best} - g^k_{best} > \epsilon$ *or* $\epsilon < \tau$ **then**
    Extend r with a new condition based on the best attribute $X_a = v_j$
    Release sufficient statistics of r
  return NR

best condition for such class. Nevertheless, it is allowed only when the rule has already been expanded with the original class at that call of `ExpandRule`. This setting is able to produce more rules in one call of `ExpandRule` and the number of rules induced from one rule r in RS in such call is at most $|C_r|$. Should the same rule already exist in the set, duplicate rule is not allowed to be expanded with given condition $X_a = v_j$. This process creates multiple rules for different classes marked as positive, but not necessarily for all the available classes in one call.

3.3 Rule Reaction to a Drift

As opposed to decision trees the set of rules offers the possibility to remove individual rules, which do not perform well, without the need of rebuilding the model. This characteristic is very important for incremental learning where change might occur, because it allows faster adaptation of the model.

The previous algorithm `VFDR` was adapting only implicitly by inferring new rules and specializing the existing ones. `AVFDR` extends the algorithm with explicit drift detection. In addition to $\mathcal{L}_r$, each rule r in `AVFDR` contains a drift detection method which tracks the performance of the rule r during learning. The method employed in `AVFDR` is the SPC [20] described in Algorithm 2. With every labeled training example covered by a rule, the rule makes prediction and updates its error rate. `SPC` monitors the error rate and manages two registers during training: p_{min} and s_{min}, where p is error rate and s is standard deviation. Every time a new example $(\boldsymbol{x}_n, y_n)$ is covered by the rule those values are updated when $p_n + s_n$ is lower than $p_{min} + s_{min}$.

The learning process of given rule can be in one of the following 3 stages: *in-control*, *out-of-control*, or in *warning*. We follow the 3-sigma rule [21]: the *warning* stage is reached if $p_n + s_n \geq p_{min} + \beta \times s_{min}$ and the *out-of-control* stage is reached if $p_n + s_n \geq p_{min} + \alpha \times s_{min}$, where $\alpha = 3$ and $\beta = 2$.

Algorithm 5: `ExpandEmptyRule`: Expanding Empty Rule.

input : r: One Rule;
δ: Confidence
τ: Constant to solve ties
output: NR: New Rules Set;
begin
  $NR \leftarrow \{r\}$
  Compute $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$ (Hoeffding bound)
  foreach *class* c_k *in ascending order of class frequency* **do**
    $EvaluateLiterals(c_k)$
    if $g^k_{best} - g^k_{2best} > \epsilon$ *or* $\epsilon < \tau$ **then**
      create new r' as $X_a = v_j \Rightarrow c_k$
      $NR \leftarrow NR \cup \{r'\}$
  if NR *is not empty* **then**
    Release sufficient statistics of r
  return NR

In warning stage the rule stops learning until either the status if the rule becomes again *in-control*. If the rule reaches *out-of-control* it implies that the performance of the particular rule has degraded significantly and can negatively influence the quality of predictions of the classifier therefore it is removed from the set. This control enables to keep the set up-to-date and prevent the set from excessive growth.

3.4 Classification Strategy

The set of rules learned by `AVFDR` can employ different classification strategies: First Hit, Weighted Max, and Weighted Sum. The ordered rules use First Hit strategy as the most appropriate. For the unordered rules we decided to apply Weighted Sum in the rest of the paper. In that case all rules covering the example are used for classification and the final class is decided by using weighted vote.

More specifically, assume that a rule r covers a test example. The example will be classified using the information in $\mathcal{L}_r$ of that rule. The simplest strategy uses the distribution of the classes stored in $\mathcal{L}_r$, and classify the example in the class that maximizes $p(c_k)$. This strategy only uses the information about class distributions and does not look for the attribute-values; therefore it uses only a small part of the available information. In a more informed strategy, a test example is classified with the class that maximizes the posteriori probability given by Bayes rule assuming the independence of the attributes given the class. There is a simple motivation for this option. $\mathcal{L}$ stores information about the distribution of the attributes given the class usually for hundreds or even thousands of examples, before expanding the rule and re-initializing the counters. Naive Bayes (`NB`) takes into account not only the prior distribution of the classes, but also the conditional probabilities of the attribute-values given the class. This way, there is a much better exploitation of the available information

Algorithm 6: `EvaluateLiterals`

input : G: Gain evaluation function;
c_k: Class

begin
  foreach *attribute* X_i **do**
    Let g_{ijk} be the $G()$ of the best literal based on attribute X_i and value v_j for class c_k
    if $g_{ijk} > g^k_{best}$ **then**
      Let $g^k_{2best} \leftarrow g^k_{best}$
      Let $g^k_{best} \leftarrow g_{ijk}$
    else
      if $g_{ijk} > g^k_{2best}$ **then**
        Let $g^k_{2best} \leftarrow g_{ijk}$

in each rule. Given the example $\boldsymbol{x} = (x_1, \ldots, x_j)$ and applying Bayes theorem, we obtain: $P(c_k|\boldsymbol{x}) \propto P(c_k) \prod P(x_j|c_k)$.

Using NB[1] in VFDT like algorithms, is a well-known technique since it was introduced by [17]. One of its greatest advantages is the boost in any-time learning property because even though the learned rule set might not be robust enough or the individual rules might not provide sufficient information for expert interpretation (not being specialized enough, i.e., having only one or few conditions), it may already be able highly informed predictions based on NB classification.

4 Experimental Evaluation

In this section we test AVFDR on datasets with different properties such that we can evaluate the behavior in various situations. We also provide comparison against decision rules which does not have explicit drift detection, and against stream decision tree classifiers[2] - VFDTc and Hoeffding Tree with ADWIN.

As an evaluation method we employed prequential classification scenario [18], where a classifier first makes a prediction and consequently the class label is observed, the error-rate is updated, and classifier is trained with said example. All algorithms were implemented in Java as an extension for KNIME [2] except ADWIN, which is implemented in MOA [3]. The evaluation datasets include both artificial and real data, as well as sets with categorical and continuous attributes or their combination.

[1] Other Bayesian techniques might be used. We use NB due to simplicity and being incremental.

[2] Default parameters for tests: N_{nb} (threshold to use NB) = 50, w_s (minimal weight of new condition partition - in rules only) = 0.1, δ (confidence level) in HB = 10^{-6}, N_{min} (examples after which HB is computed) = 200, τ (tie breaking constant) = 0.05. Due to space limitation we do not provide thorough discussion about the influence of different parameter settings.

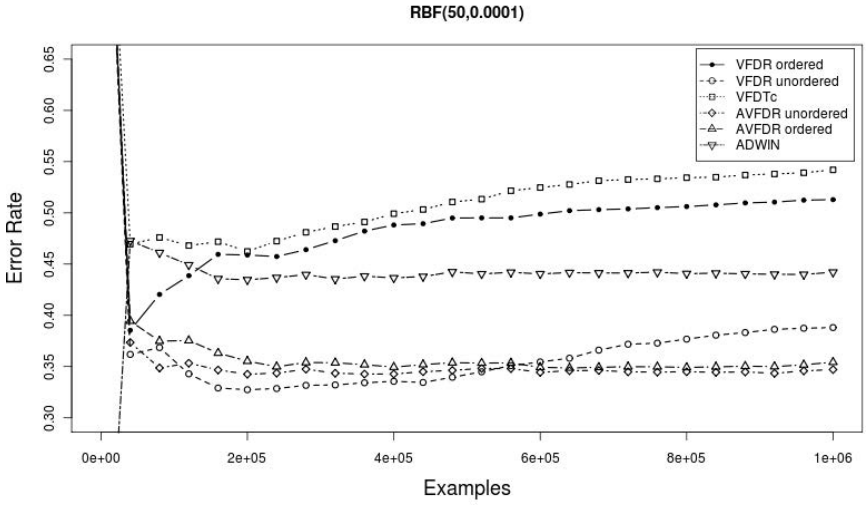


Fig. 1. The prequential error during the learning process in RBF(50,0.0001) dataset with gradual drift. Non-adaptive methods are slowly increasing their error rate.

4.1 Artificial Datasets

The artificial datasets were obtained using generators in [3].

The **Hyperplane** is generated such that the class is given by rotating hyperplane [23]. This set with 1,000,000 examples has 2 classes, 10 attributes changing at speed 0.001 with 5% noise (probability for each instance to have its class inverted). Another artificial dataset is **SEA** Concepts [32] and is commonly used in stream mining tasks that require time changing qualities of data. It is a two-class problem, defined by three attributes (two relevant) and 10 % of noise (the same as previous). There are four concepts, each containing 15,000 examples. The total number of examples is 60,000. **LED** is formed by examples [6] with $\{0, 1\}$ values of each attribute signalling whether given LED is off or on. Only seven out of 24 are relevant. Class label reflects the number (0 to 9) displayed by the relevant diodes. There is 10 % of noise added to this dataset (probability for each attribute that it would have its value inverted). The generated training set size is 1,000,000 instances. The **RBF**'s are generated datasets with drift introduced by moving centroids with constant speed. The sets have 1,000,000 examples and are described as RBF(x, y) where x is the number of centroids moving at the speed y. In **Waveform** dataset [6] there are 3 classes of waves each described with 40 attributes. This dataset had 100,000 instances and it is generated with 10 drift attributes occurring at 50,000.

4.2 Real Datasets

These datasets are mostly obtained from [14] unless cited otherwise.

The **Intrusion** detection from KDDCUP 99, is a data set describing connections which are labeled either as normal or one of four categories of attack. The dataset

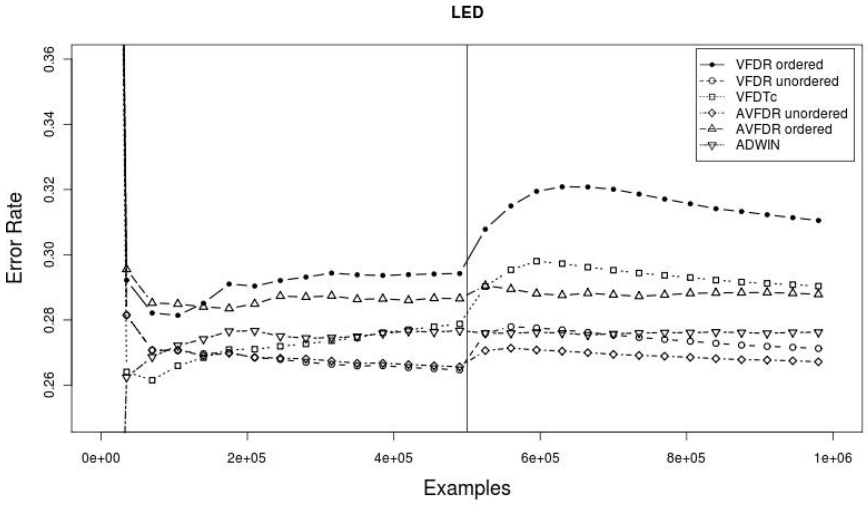


Fig. 2. The prequential error evolution throughout the learning process in LED dataset with abrupt drift. The increase of error rate after abrupt drift occurring at 500,000 is visibly larger in non-adaptive methods.

consists of 4,898,431 instances. The next set is **Covtype**, which has 54 cartographic attributes, continuous and categorical. The goal is to predict the forest cover type for given area. The dataset contains 581,012 instances. The **Elec** dataset contains data collected from electricity market of New South Wales, Australia. It has 45,312 instances. Inspired by a regression dataset used in [24], the task of **Airlines** dataset [3] is to predict whether a flight will be delayed given the information of the scheduled departure in 7 attributes. It consists of 539,383 instances. The **Connect-4** dataset from UCI repository consists of 42 categorical attributes and contains 67,557 examples. The **Activity** dataset is the Localization Data for Person Activity. The set has 164,860 instances and 8 attributes, but we removed the time information (timestamp and date).

4.3 Results

We conducted 5 runs over the synthetic datasets using different random seeds. These datasets are designed to contain concept drift therefore provide evidence of the benefit of adaptation. Figure 1 and Fig. 2 depict examples of the performance of the classifiers in selected artificial datasets. The first one, Figure 1, shows RBF(50,0.0001) dataset, which contains gradual concept drift. It can be observed that both AVFDRs cope best with the drift. ADWIN contains adaptation technique to adapt to drift and sustains its accuracy as opposed to VFDT and VFDRs which over time decrease their respective accuracies. In the second example, LED dataset on Figure 2, the sudden concept drift is highlighted by vertical line at 500,000 where all classifiers except ADWIN have noticeably increased error-rate. The increase is lower in the case of adaptive methods. AVFDR^u has the best reaction and ADWIN keeps its performance on very stable level.

Table 1. Prequential error rates of rule based classifiers using NB. For comparative purposes, we report the prequential error of VFDTc using NB classifiers at the leaves.

	Error rate % (variance)					
	VFDR^o	VFDR^u	VFDTc	AVFDR^u	AVFDR^o	ADWIN
LED	30.67 (0.53)	27.17 (0.06)	29.35 (0.11)	**26.65 (0.01)**	28.79 (0.04)	27.63 (0.00)
RBF(0,0)	20.43 (2.62)	**11.90 (2.48)**	18.6 (3.74)	12.07 (2.1)	21.99 (0.63)	19.05 (4.70)
RBF(50,0.001)	68.74 (6.48)	62.15 (0.83)	69.37 (4.93)	57.63 (19.05)	60.27 (2.34)	**52.50 (13.1)**
RBF(50,0.0001)	49.74 (1.19)	37.62 (2.82)	54.54 (2.10)	**33.60 (0.86)**	34.93 (2.18)	43.81 (1.15)
SEA	15.64 (0.34)	14.68 (0.31)	15.51 (0.13)	14.14 (0.41)	15.6 (0.03)	**13.48 (0.08)**
Hyperplane	14.63 (0.47)	13.24 (0.53)	15.01 (0.76)	**12.64 (0.31)**	14.72 (0.43)	12.76 (0.18)
Waveform	24.43 (0.38)	19.07 (0.35)	20.27 (1.07)	**18.90 (0.61)**	23.40 (013)	19.96 (0.02)
Intrusion	7.0E-4	**2.9E-4**	5.4E-4	4.3E-4	5.0E-4	0.15
Covtype	18.88	**13.65**	15.21	15.34	15.35	19.53
Elec	28.75	25.53	27.58	25.37	24.38	**19.56**
Airlines	33.46	**32.46**	33.67	32.94	33.41	37.33
Connect-4	27.67	27.73	**26.41**	26.86	27.38	27.96
Activity	34.47	27.16	38.54	**17.03**	18.91	42.48
Average rank	5	2.54	4.31	1.78	3.62	3.77

Using artificial datasets with known position of drifts is especially beneficial for closer examination of the drift detection. In the case of SEA datasets we observed for AVFDR^u that first drifts appeared on average after 570, 502 and 285 examples of the second, third and fourth concept respectively. This reflects the increasing difference between the concepts. There were various numbers of other drifts following with various delays, which is expected since a rule reflects only local change of the feature space and might react much later depending on the examples arriving. This is a good result. A standard NB with SPC, a very good classifier for this task, needed on average 1328, 1024 and 461 examples to detect the (global) change. AVFDR^o was not that successful. In some cases the some drifts were not detected at all or very late. LED datasets were also generated with known sudden concept drift. The results are not that straightforward. The noise, irrelevant attributes and relatively high number of classes in combination with higher sensitivity in local detection caused that many rules, that were generated, were pruned before they were precise enough. Besides these false drifts many of the actual local drifts were detected mostly between 30-100 examples.

The second group of datasets is real world data. The presence of concept drift is unknown but can be expected. In these datasets, VFDR^u mostly performs well since change, if there is any, is probably very slow. Nevertheless, the size of the decision model grows one order of magnitude larger than the other rule learning systems. AVFDR^u still presents very competitive performance in most of the real world sets with the advantage of smaller rule set size. The example of prequential error evolution throughout the learning process in real world dataset is plotted on Fig. 3. The noticeably worst performing classifier in Airlines dataset is ADWIN. The other classifiers perform relatively similarly with the unordered rules sets being slightly better than the rest.

The results from all the tests are collected in Tab. 1 with averaged score ranks. In order to compare multiple classifiers on multiple datasets we applied the Friedman test [31] and post-hoc Nemenyi test [10], based on average ranks achieved on all the datasets, with p-value of 0.05. At this level AVFDR^u was significantly better than VFDR^o

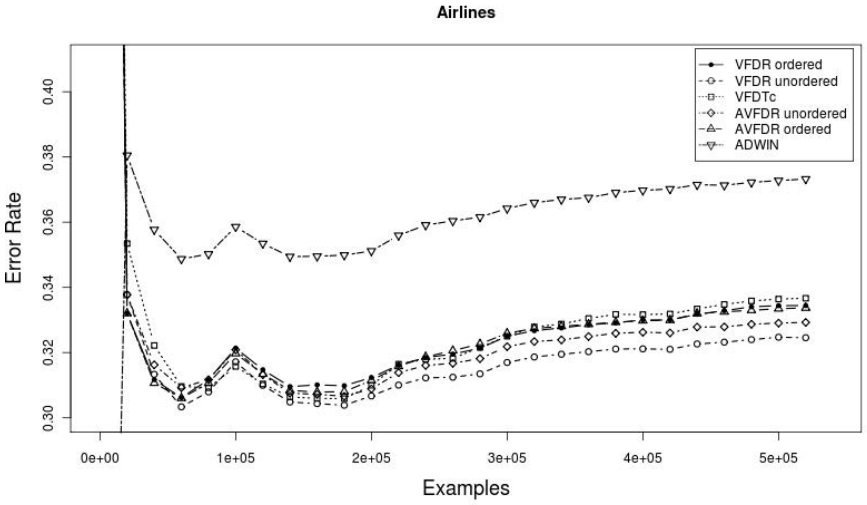


Fig. 3. The prequential error evolution throughout the learning process in real-world Airlines dataset where the presence of drift is not known

and VFDTc. VFDR^u was significantly better than VFDR^o. With p-value of 0.1 AVFDR^u was also significantly better than ADWIN.

Conceptually, each rule is closely related to a path in a decision tree. The consequent of a rule plays similar role as a leaf. They contain practically the same information and possess the same functionality. Therefore, the size of the classifiers can be compared by considering the number of rules in a set and number of leaves in a tree. The Table 2 shows the number of rules/leaves (the average number in case of the artificial datasets) that were available at the end of the prequential process. Note that among these classifiers only AVFDRs are able to decrease the number over the learning process. VFDR^u produces the largest classifier, which is expected due to its design. But it can be observed that by using the drift detecting technique AVFDR is able to effectively reduce the set while having good accuracy. Only in Covtype, complex real dataset, the much larger set of VFDR^u achieved notably better result.

In order to analyze the time we provide the prediction and learning times in prequential evaluation process of the classifiers in relation with the learning time of VFDTc in the right part of Tab. 2. Since ADWIN was running on different framework it is not included in this evaluation. We can conclude that the times of ordered rule sets are practically the same as in case of VFDTc. For the unordered rules of VFDR^u the time is dependent on the number of rules, because as opposed to tree search the whole rule set is scanned for rules that cover an example. The advantage of AVFDR is that it removes potentially incorrect rules thus keeps the set smaller and learning and prediction times are reduced.

Table 2. The number of rules in decision rule learners at the end of the learning process. For comparative purposes we present also the number of leaves generated by VFDTc. Relative times refer to the ratio of the learning time of rule learners with respect to the learning time of VFDTc.

	Size of Decision Models					Relative times			
	VFDR^o	VFDR^u	VFDTc	AVFDR^u	AVFDR^o	VFDR^o	VFDR^u	AVFDR^u	AVFDR^o
LED	26	3698	50	972	16	1.1	14.5	11.9	1.5
RBF(0,0)	33	2012	81	1255	22	0.8	14.2	11.7	1.1
RBF(50,0.001)	21	6029	52	1	2	1.3	52.3	1.7	1.3
RBF(50,0.0001)	56	745	80	6	1	0.9	9.8	0.9	0.9
SEA	13	68	28	35	12	1.1	1.5	1.1	1
Hyperplane	79	1348	353	514	62	1.2	12.2	5.4	1.2
Waveform	13	637	12	129	4	0.8	9.5	2.7	0.9
Intrusion	28	424	642	5	5	1	6.6	0.9	1.1
Covtype	79	844	393	23	16	1.4	7.8	1.3	0.9
Elec	22	71	38	4	6	1.1	2.7	0.9	0.5
Airlines	59	501	1096	157	34	1.1	2.7	0.9	0.5
Connect-4	14	21	31	7	4	1	1.4	0.9	1.2
Activity	61	4278	85	31	7	1	36.9	0.9	0.8

5 Conclusions

This work presented Adaptive Very Fast Decision Rule classifier, which incrementally induces rules for each class in a decision problem. It requires only one scan of data and provides any-time classification model that is capable of fast adaptation to changes in data. The adaptation is achieved by exploiting the modularity and independence of single rules within the rule set and assigns an error based on a drift detection method to each rule. Whenever the quality of a rule decreases significantly, the rule is removed from the set. This approach works not only as fast adaptation tool but also as rule set pruning method.

The proposed AVFDR method was tested on multiple artificial and real-world datasets and achieved very promising results. Especially the unordered version, AVFDR^u, was very successful in fast adaptation in artificial sets with known drift. In all the datasets AVFDR^u achieved very good ranks and was significantly better than state-of-the-art Hoeffding Tree algorithm VFDTc. The extension effectively reduced the rule set size for time changing data streams which also reduced the learning and prediction in prequential evaluation process.

Acknowledgments. This work is funded by the ERDF - through the COMPETE programme and by National Funds through the FCT Project KDUS (PTDC/EIA/098355/2008). Petr Kosina acknowledges the support of Fac. of Informatics, MU, Brno.

References

1. Baena-Garcia, M., Campo-Avila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R.: Early drift detection method. In: 4th International Workshop on Knowledge Discovery from Data Streams, ECML-PKDD, Berlin, Germany (2006)

2. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., Wiswedel, B.: Knime - the konstanz information miner: version 2.0 and beyond. SIGKDD Explor. Newsl. 11, 26–31 (2009)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. Journal of Machine Learning Research, JMLR (2010)
4. Bifet, A., Gavaldà, R.: Adaptive Learning from Evolving Data Streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009)
5. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 139–148. ACM, New York (2009)
6. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees, 1st edn. Chapman and Hall/CRC (1984)
7. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning 3, 261–283 (1989)
8. Clark, P., Boswell, R.: Rule induction with CN2: Some recent improvements. pp. 151–163. Springer (1991)
9. Cohen, W.: Fast effective rule induction. In: Prieditis, A., Russel, S. (eds.) Machine Learning, Proc. of the 12th International Conference. Morgan Kaufmann (1995)
10. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
11. Domingos, P.: Unifying instance-based and rule-based induction. Machine Learning 24, 141–168 (1996)
12. Domingos, P.: Mining high-speed data streams. pp. 71–80. ACM Press (2000)
13. Ferrer, F., Aguilar, J., Riquelme, J.: Incremental rule learning and border examples selection from numerical data streams. Journal of Universal Computer Science 11(8), 1426–1439 (2005)
14. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
15. Gama, J., Fernandes, R., Rocha, R.: Decision trees for mining data streams. Intelligent Data Analysis 10, 23–45 (2006)
16. Gama, J., Kosina, P.: Learning decision rules from data streams. In: Proc. of the 22nd International Joint Conference on Artificial Intelligence, pp. 1255–1260. AAAI, Menlo Park (2011)
17. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: Proc. of the 9th International Conference on Knowledge Discovery and Data Mining, ACM Press, New York (2003)
18. Gama, J., Sebastiäo, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 329–338. ACM, New York (2009)
19. Gama, J.: Knowledge Discovery from Data Streams. Chapman and Hall/CRC (2010)
20. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with Drift Detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004)
21. E., Grant, Leavenworth, R.: Statistical Quality Control. McGraw-Hill (1996)
22. Hinkley, D.: Inference about the change point from cumulative sum-tests. Biometrika 58, 509–523 (1970)
23. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 97–106. ACM, New York (2001)
24. Ikonomovska, E., Gama, J., Džeroski, S.: Learning model trees from evolving data streams. Data Min. Knowl. Discov. 23, 128–168 (2011)

25. Klinkenberg, R.: Learning drifting concepts: Example selection vs. example weighting. Intelligent Data Analysis 8(3), 281–300 (2004)
26. Kosina, P., Gama, J.: Very fast decision rules for multi-class problems. In: Proc. of the 2012 ACM Symposium on Applied Computing, pp. 795–800. ACM, New York (2012)
27. Maloof, M., Michalski, R.: Incremental learning with partial instance memory. Artificial Intelligence 154, 95–126 (2004)
28. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993)
29. Rivest, R.: Learning decision lists. Machine Learning 2, 229–246 (1987)
30. Schlimmer, J.C., Granger, R.H.: Incremental learning from noisy data. Machine Learning 1, 317–354 (1986)
31. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures, 4th edn. Chapman & Hall/CRC (2007)
32. Street, W.N., Kim, Y.: A streaming ensemble algorithm SEA for large-scale classification, pp. 377–382. ACM Press (2001)
33. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–235. ACM Press (2003)
34. Weiss, S., Indurkhya, N.: Predictive Data Mining, a practical Guide. Morgan Kaufmann Publishers (1998)
35. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23, 69–101 (1996)

Improved Counter Based Algorithms for Frequent Pairs Mining in Transactional Data Streams

Konstantin Kutzkov

IT University of Copenhagen
konk@itu.dk

Abstract. A straightforward approach to frequent pairs mining in transactional streams is to generate all pairs occurring in transactions and apply a frequent items mining algorithm to the resulting stream. The well-known counter based algorithms FREQUENT and SPACE-SAVING are known to achieve a very good approximation when the frequencies of the items in the stream adhere to a skewed distribution.

Motivated by observations on real datasets, we present a general technique for applying FREQUENT and SPACE-SAVING to transactional data streams for the case when the transactions considerably vary in their lengths. Despite of its simplicity, we show through extensive experiments that our approach is considerably more efficient and precise than the naïve application of FREQUENT and SPACE-SAVING.

1 Introduction

Mining heavy items from data streams is a fundamental problem in knowledge discovery. It has been widely studied from both theoretical and practical point of view, see [10] for an overview of achieved results. Motivated by applications for finding associations among purchased items in market baskets, many algorithms have been developed for mining frequent k-itemsets, for $k \geq 2$. In the case when transactions are revealed one at a time in a streaming fashion, a straightforward approach to mining the frequent k-itemsets is to generate all size-k subsets in each transaction and then apply a frequent items mining algorithm to the resulting stream. However, this approach does not make use of the additional information that for a transaction of length ℓ we implicitly know the next $\binom{\ell}{k}$ k-itemsets in the stream. In this paper we show that for many real datasets this information allows us to design significantly more efficient and robust algorithms than the naïve application of counter-based frequent items mining algorithm.

Transactional data streams. Classic algorithms like Apriori [1] and FP-Growth [14] need several passes over the input to find the frequent itemsets. This implies that we need a persistently stored database and for many applications this is not a feasible requirement. Historically, Manku and Motwani [25] first recognized the necessity for frequent itemset mining algorithms over high-speed transactional data

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 843–858, 2012.

streams. They heuristically generalized their LOSSY COUNTING algorithm and observed that it provides good estimates for the most frequent itemsets. Association rule mining from transactional data streams has been recognized a major research problem due to continuously increasing data volume, see [15] for an overview of the area.

Assuming that transactions are either generated by a random process or arrive in a random order, researchers have designed randomized algorithms for frequent itemset mining [8,30]. The theoretical analysis of the algorithms performance thus utilizes Chernoff bounds in order to show quality estimates with high probability. However, experiments indicate [8] that such assumptions are too optimistic for real datasets and the results are not nearly as good as suggested by the theoretical analysis.

Frequent items mining. Algorithms for frequent items mining in data streams can be roughly divided in two categories: sketch-based and counter-based algorithms. Sketch-based algorithms work by hashing the items to a small sketch of the data stream processed so far and updating a corresponding counter. The frequency of individual items can be then estimated by reading a counter in the sketch. The two well-known algorithms COUNT-SKETCH [9] and COUNT-MIN [11] are based on this approach. The sketches consist of $O(1/\varepsilon)$ counters. COUNT-SKETCH provides an additive approximation of $O((\varepsilon F_2)^{\frac{1}{2}})$ and COUNT-MIN of $O(\varepsilon F_1)$ where F_p is the p-norm of the frequency vector of the stream. In order to guarantee that the approximation is correct with high probability, one runs several copies of the algorithm in parallel and returns the median of the estimates.

Counter based algorithms are deterministic. They maintain a summary of the items processed so far. The summary consists of a small subset of the items with associated counters approximating the frequency of the item in the stream. For a summary maintaining $O(1/\varepsilon)$ entries, they provide an additive approximation εF_1. It was experimentally observed that counter based algorithms provide better guarantees than sketch based algorithms but the reasons have been unclear. In a recent work Berinde et al. [4] present an analysis of so called *heavy tolerant counter based* algorithms, including FREQUENT [13,17,27] and SPACE-SAVING [26]. They show that both algorithms are clearly superior to sketch based algorithms, see the third paragraph of Section 2 for a detailed discussion.

Mining frequent k-itemsets. In this paper we consider frequent pairs mining in transactional data streams. Our algorithm can be extended in a straightforward way to k-itemsets mining but for the ease of presentation we concentrate on results for frequent pairs mining. Note that the generation of the frequent pairs is considered to be the most time consuming phase in Apriori [1]. As noted by Park et al. [28] *"...the initial candidate set generation, especially for the large 2-itemsets, is the key issue to improve the performance of data mining".*

Our contribution. The straightforward application of frequent items mining algorithms is the only known approach to mining frequent itemsets from transactional data streams with rigorously understood behaviour. The focus in our

work is on improving the approach for counter based frequent items algorithms. Our main contributions can be summarized as follows:

- We make the observation that for many real-life datasets the length of the transactions considerably vary and most of the pairs in the transactional stream are generated by a fraction of the transactions.
- Utilizing the main idea behind the FREQUENT algorithm, we present a simple technique for adjusting FREQUENT and SPACE-SAVING to mining frequent pairs in transactional data streams.
- We prove that our modified approach yields an additive approximation error at least as good as the original algorithms. However, through extensive experiments on real datasets, we obtain considerable improvements in both running time and accuracy of the estimates.

Building upon FREQUENT, Jin and Agrawal [16] presented a method for improving the space requirements of Apriori. Thus, our algorithm automatically yields an improved version of Apriori for the considered class of datasets.

2 Preliminaries

Notation. Let $\mathcal{I}$ be a set of n items. We assume a total order on $\mathcal{I}$ and for the ease of presentation we set $\mathcal{I} = \{0, 1, 2, \ldots, n-1\}$. A *weighted stream* $\mathcal{S}$ over $\mathcal{I}$ is a sequence of m entries (i, v) for an item $i \in \mathcal{I}$ and weight $v \in \mathbb{R}^+$. When for all entries (i, v) in $\mathcal{S}$, $v = 1$ holds, we will refer to $\mathcal{S}$ as an *unweighted* stream. A *transaction* T is a subset of items of $\mathcal{I}$. The *length* of a transaction T, denoted as $|T|$, is the number of items occurring in it. A *transactional stream* $\mathcal{T}$ consists of m transactions revealed one at a time, $p = (i, j) \subset \mathcal{I}$ for $i < j$ is a *pair*.

The *frequency* or *weight* of an item i is defined as $f_i = \sum_{(i,v)\in\mathcal{S}} v$ and $\mathbf{f}_\mathcal{S} = (f_0, \ldots, f_{n-1})$ is the *frequency vector* of the stream $\mathcal{S}$. For unweighted streams the frequency is simply the number of occurrences of i in $\mathcal{S}$. For transactional streams the frequency of a pair p is the number of transactions containing p, $|\{T_j : p \subseteq T_j\}|, 1 \leq j \leq m$. The ℓ-norm of a weighted stream $\mathcal{S}$ over a set $\mathcal{I}$ is denoted as $F_\ell(\mathcal{S}) = (\sum_{i\in\mathcal{I}} f_i^\ell)^{\frac{1}{\ell}}$ for $\ell \geq 0$. For an unweighted stream $\mathcal{S}$, $F_1(\mathcal{S})$ is simply the length, i.e. the number of entries, in $\mathcal{S}$. Without loss of generality, we assume that the items $i \in \mathcal{I}$ are ordered by decreasing frequency such that $f_1 \geq f_2 \geq \ldots \geq f_n$. The kth item in this sequence is the item *of rank* k. The *k-residual ℓ-norm* of a stream $\mathcal{S}$ is then defined as $F_\ell^{res(k)}(\mathcal{S}) = (\sum_{i=k+1}^{n} f_i^\ell)^{\frac{1}{\ell}}$. Clearly, $F_\ell^{res(0)}(\mathcal{S}) = F_\ell(\mathcal{S})$. Items with weight above εF_1, for a user-defined $0 < \varepsilon < 1$, will be called *ε-heavy hitters* or just *heavy hitters* when ε is clear from the context.

Heavy tolerant counter based algorithms. Let us first briefly recall how the algorithms FREQUENT and SPACE-SAVING work for an unweighted stream of items. Both algorithms maintain a summary B of b entries. The summary consists of items $i \in \mathcal{I}$ with associated counter c_i serving as an estimate of the frequency of i in the stream processed so far. We will refer to the *size of the summary* as the

number of entries that can be recorded in the summary. When a new item j arrives, we check whether j is recorded in B. If so, we increment c_j by 1. If not, we check whether all b slots are occupied by an entry and if not, we insert the entry $(j, 1)$ to B. The algorithms proceed differently when j is not recorded in B. In this case FREQUENT decrements by 1 the counters of all entries in the summary and then removes entries with a counter equal to 0. SPACE-SAVING replaces an entry (k, c_k) with the smallest counter by the new entry $(j, c_k + 1)$. After processing the stream one estimates the frequency of a given item i by checking the corresponding entry in the summary. If i is not recorded in the summary, FREQUENT returns 0 as an estimate of i's frequency and SPACE-SAVING returns minimum counter in the summary. A simple analysis shows that after processing the whole stream, FREQUENT guarantees that the frequency of each item is underestimated by at most F_1/b, and SPACE-SAVING provides an upper bound of the overestimation of each item by F_1/b. This leads to the following

Definition 1. *For a stream $\mathcal{S}$, a counter based algorithm $\mathcal{A}$ with a summary of size $b = \lceil 1/\varepsilon \rceil$ provides an* ε-heavy hitter guarantee *if $\delta_i \leq \lceil \varepsilon F_1 \rceil$ for all items $i \in \mathcal{I}$ where δ_i is the absolute additive approximation error. $\mathcal{A}$ provides then an* ε-approximation *of the weight of each item and it is called an* ε-heavy tolerant *counter based algorithm.*

The above implies that after processing a stream $\mathcal{S}$, a counter based algorithm providing an ε-heavy hitter guarantee will correctly detect all ε-heavy hitters. The algorithms can be generalized in a straightforward way to handle weighted updates such that the approximation guarantee is maintained. The generalization poses a challenge only for the processing time of each incoming item, see the discussion on the running time of our algorithm in Section 3 for more on this.

Counter based vs. sketch based frequent items mining algorithms. Skewness in the frequency distribution of items in data streams is ubiquitous, see for example [10,12]. Therefore, an algorithm providing an approximation guarantee in terms of the residual norm of a stream can be considered superior to algorithms with approximation error depending on the norm of the whole stream. The analysis of randomized sketch based algorithms naturally applies to obtaining strong k-tail approximation guarantees, i.e. depending on $F_1^{res(k)}$, for the frequency distribution of individual items. For data streams adhering to a skewed distribution such guarantees are much stronger than the heavy hitter guarantees provided by the naïve analysis of counter based analysis. Building upon work by Bose et al. [5], Berinde et al. [4] recently presented a deeper analysis of the behaviour of heavy tolerant counter-based algorithms. In particular, their results show that FREQUENT and SPACE-SAVING yield a k-tail guarantee of $(\varepsilon/k)F_1^{res(k)}$ using a summary with $O(k/\varepsilon)$ entries as opposed to sketch based algorithms requiring $O((k/\varepsilon)\log n)$ counters. This result closed the discrepancy between the better results yielded by counter based algorithms on real and synthetic data sets as compared to counter based algorithms, clearly indicating that counter based algorithms are superior.

Approximation guarantees of FREQUENT *and* SPACE-SAVING. In the following we list several results about the approximation guarantees provided by counter based algorithms. The first one is a generalization of the correctness argument of FREQUENT.

Lemma 1. *After decreasing d times the weight of at least $s > b = \frac{1}{\varepsilon}$ distinct items in a stream $\mathcal{S}$, such that no item has negative weight, all ε-heavy hitters are guaranteed to have positive weight.*

Proof. Since no item can have negative weight, $d \leq F_1(\mathcal{S})/s$ holds. This implies $f_i - d > 0$ for all $i : f_i \geq \varepsilon F_1(\mathcal{S})$.

Lemma 2. *[4,26] After processing a weighted stream $\mathcal{S}$ by either* FREQUENT *or* SPACE-SAVING *the following hold*

- *The sum of all counters in the summary of* SPACE-SAVING *is exactly $F_1(S)$.*
- *For a summary of size $b = O(k/\varepsilon)$, the underestimation of the approximation returned by* FREQUENT *of the weight of each entry is bounded by $(\varepsilon/k)F_1^{res(k)}$.*
- *For a summary of size $b = O(k/\varepsilon)$, the overestimation of the approximation returned by* SPACE-SAVING *of the weight of each entry is bounded by $(\varepsilon/k)F_1^{res(k)}$.*

From the above we derive the following

Definition 2. *A counter based algorithm with a summary of size $O(k/\varepsilon)$ provides a (k, ε)-tail guarantee if the additive approximation of the weight of each item is bounded by $(\varepsilon/k)F_1^{res(k)}$.*

3 Our Approach

Motivation. Before presenting our improved algorithm, let us give some informal motivation. Figure 1 presents the distribution of transaction lengths for the Kosarak dataset and the number of pairs generated from transactions with the given length. It turns out that more than 90% of the pairs are generated by less than 10% of the transactions. Thus, by efficiently processing the long transactions we can achieve considerable improvement in the running time of a counter based algorithm applied to a stream of pairs occurring in transactions. A naïve solution would be to simply not consider long transactions. This approach would yield incomplete results since it is often the case the long transactions contain more specific but still important information. In Section 4 we show that for datasets from various real-life domains the pairs per transaction distribution follows this pattern.

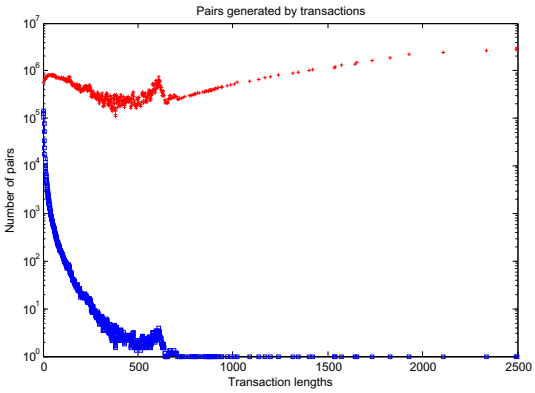


Fig. 1. The number of pairs in the Kosarak dataset generated by transactions with the given length

The basic idea. At the heart of our improved counter based algorithms is the following simple observation. Assume we want to find an ε-approximation of the frequencies of the items in a given unweighted stream. If we repeatedly remove $s \geq 1/\varepsilon$ distinct items from the stream, and then apply a counter based frequent items mining algorithm to the updated stream, we will again achieve an ε-approximation for all items in the stream. An informal correctness argument is that each item removal is witnessed by the removal of $s-1$ other distinct items and thus the total decrease in the item frequency is bounded by $F_1/s \leq \varepsilon F_1$, thus the heavy hitter guarantee is maintained. Essentially, this is a reformulation of the main idea of the FREQUENT algorithm.

For mining frequent pairs in transactional streams the above idea may be useful because we know that a transaction of length ℓ contains $\binom{\ell}{2}$ distinct pairs. This suggests a more efficient way for processing the long transactions.

The Algorithm

A high-level pseudocode description of our algorithm FREQUENTPAIRSMINER is presented in Figure 2. The input consists of a counter based algorithm $\mathcal{A}$ providing an (ε, k)-approximation, a stream of transactions $\mathcal{T}$ and a user-defined parameter t. We will distinguish between the cases when $\mathcal{A}$ is either FREQUENT or SPACE-SAVING. $\mathcal{A}$ maintains a summary B consisting of b entries. We assume that $\mathcal{A}$ is capable of handling weighted updates. We proceed a transaction $T \in \mathcal{T}$ as follows: If $|T| \leq t$ we generate all $\binom{t}{2}$ pairs occurring in T and feed them into $\mathcal{A}$. Otherwise if $|T| > t$ we add T to a batch of long transactions $\mathcal{L}$ and check whether the number of distinct pairs in $\mathcal{L}$ is bigger than ℓ/ε for $\ell \geq 1$. If so, we find the number of occurrences of all pairs occurring more than ℓ times in $\mathcal{L}$, decrease their weight by ℓ, and feed the resulting weighted stream into $\mathcal{A}$.

function FREQUENTPAIRSMINER
Input: stream of transactions $\mathcal{T}$, a counter based algorithm $\mathcal{A}$, a threshold t
1: **for** each transaction $T \in \mathcal{T}$ **do**
2: **if** $|T| \leq t$ **then**
3: Generate the set $\mathcal{P}_T$ of all pairs in T
4: **for** each pair $p \in \mathcal{P}_T$ **do**
5: call $\mathcal{A}.update(B, p, 1)$
6: **else**
7: add T to a batch $\mathcal{L}$
8: **if** there are ℓ groups of at least b distinct pairs in $\mathcal{L}$ **then**
9: Compute the set $P_{\mathcal{L}}$ of weighted pairs occurring in $\mathcal{L}$ more than ℓ times
10: **for** each $(p, w_p) \in P_{\mathcal{L}}$ **do**
11: call $\mathcal{A}.update(B, p, w_p - \ell)$
12: Report all pairs in the summary B and their estimated frequency.

Fig. 2. A high-level pseudocode description of the algorithm

Theorem 1. *Let $\mathcal{T}$ be a transactional stream and $\mathcal{A} = \{$*FREQUENT, SPACE-SAVING*$\}$ with a summary of size $O(k/\varepsilon)$. Then for the stream of pairs in $\mathcal{T}$* FREQUENTPAIRSMINER *provides a (k, ε)-tail guarantee.*

Proof. We first show that FREQUENTPAIRSMINER provides an ε-approximation. Assume that d times we eliminate at least $1/\varepsilon$ distinct pairs in lines 8-11 and decrease the weight of a pair by at most d. Then, we feed $\mathcal{A}$ with a stream with $F_1 - d/\varepsilon$ pairs. $\mathcal{A}$ provides an ε-approximation, thus we have that the absolute additive approximation for each pair is bounded by $\varepsilon(F_1 - d/\varepsilon) + d = \varepsilon F_1$.

Next we prove that if δ is an upper bound on the absolute additive approximation, then $\frac{\delta k + F_1^{res(k)}}{b}$ is also an upper bound on the additive approximation of each pair for any $0 \leq k \leq P$ where P is the total number of distinct pairs in the stream. We consider two cases:

1. $\mathcal{A}$ =FREQUENT. In this case we can charge the underestimation only from pairs occurrences that have been deleted from the stream. Clearly, their number is upper-bounded by $k\delta + F_1^{res(k)}$ and since each deletion is witnessed by the deletion of at least $b + 1$ distinct pairs, the bound follows.
2. $\mathcal{A}$ =SPACE-SAVING. In the following we denote by $\mathcal{S}$ the stream of pairs generated from $\mathcal{T}$. Let $\mathcal{P}^k$ be the set of the k most frequent pairs in the transactional data stream. The frequency of each pair $p \in \mathcal{P}^k$ is approximated within absolute additive error of δ. Denote by $\mathcal{S}_{\mathcal{A}}$ the stream fed into $\mathcal{A}$ and by $\mathcal{S}_{\mathcal{L}}$ the stream of the remaining pairs in the transactional stream $\mathcal{S}$. By Lemma 2 the total sum of counters in the summary of $\mathcal{A}$ is then $F_1(\mathcal{S}_{\mathcal{A}})$. Now consider the k counters in the summary of $\mathcal{A}$ with the largest value. Each of them approximates a pair with an additive error at most δ. Also, the ith such counter, $i \leq k$, has a value of at least the weight of the ith heaviest pair in $\mathcal{S}_{\mathcal{A}}$, which in turn implies that its value is at least the weight of the

ith heaviest pair in $\mathcal{S}$ restricted to $\mathcal{S}_\mathcal{A}$. Therefore, together with Lemma 2 we obtain that the total sum of the counters in the summary, different from the largest k counters, is bounded by $F_1^{res(k)}(\mathcal{S}_\mathcal{A})$. Also, we have deleted $F_1(\mathcal{S}_\mathcal{L})$ pairs occurrences from $\mathcal{S}$. Therefore, the total approximation error for all pairs amounts to $k\delta + F_1^{res(k)}(\mathcal{S}_\mathcal{A}) + F_1^{res(k)}(\mathcal{S}_\mathcal{L})$. By Lemma 2 the overestimation of each pair in the stream $\mathcal{S}_\mathcal{A}$ is bounded by $\frac{k\delta+F_1^{res(k)}(\mathcal{S}_\mathcal{A})}{b}$. In the stream $\mathcal{S}_\mathcal{L}$ we delete d times at least b distinct pairs, thus together with Lemma 1 we obtain that we can charge to each pair an approximation error of at most $\frac{k\delta+F_1^{res(k)}(S_\mathcal{A})+F_1^{res(k)}(S_\mathcal{L})}{b} \leq \frac{k\delta+F_1^{res(k)}}{b}$.

Once we have the recursive form for the upper bound, we can continue iterating in this way setting $\delta_i = \frac{k\delta_{i-1}+F_1^{res(k)}(\mathcal{S})}{b}$ while $\delta_i \leq \delta_{i-1}$ for the approximation error in the ith step holds. We either reach a state where no progress is made or, since the approximation error is lower bounded by $F_1^{res(k)}(\mathcal{S})$, we have $\delta_i \to \delta$ as $i \to \infty$. In either case the claim follows. Setting $b = O(k/\varepsilon)$ concludes the proof. □

Running time. We present efficient algorithms for the steps in FREQUENT-PAIRSMINER. We assume that the summary can be updated in constant time. For FREQUENT a hashtable implementation guarantees a constant amortized processing time per pair. More sophisticated data structures [13,17] improve this to constant time in the worst case. For SPACE-SAVING a linked list implementation of the summary [26] guarantees constant worst case processing time per pair. Using the modification from [7] one can achieve constant processing time for weighted pairs such that the approximation error is increased by at most a factor of 2. Using similar techniques one can obtain constant time for weighted updates for FREQUENT.

The running time crucially depends on how we process a batch of long transactions and mine the frequent pairs in the batch. Under the assumption that many items will occur only once in the batch a suitable choice would be a classic frequent itemset mining algorithm like Apriori [1] or FP-growth [14]. However, in this case we have a relatively small number of long transactions. Under the assumption that most pairs will occur at most once in the batch, we present another approach aimed at minimizing the running time:

For each combination of two transactions in the batch $\mathcal{L}$ we compute their intersection, generate the pairs occurring in the intersection and store them in a hashtable associating a set of transactions IDs with each pair. Assuming transactions are given in sorted order we can compute the intersection of two transactions in linear time. Therefore, for a batch of q transactions, such that the total number of items in the transactions in the batch is r, the running time is upper bounded by $O(qr)$: for each transactions we compute in time $O(r)$ the intersections with the other transactions. Let $\mathcal{P}^\mathcal{L}$ be the pairs occurring in $\mathcal{L}$ and $f^\mathcal{L}$ be the frequency vector of $\mathcal{P}^\mathcal{L}$. Assuming a given pair p occurs $f^\mathcal{L}(p)$ times in the batch $\mathcal{L}$, it will occur in $\binom{f^\mathcal{L}(p)}{2}$ transaction intersections. Then the number of pairs generated from all transactions is bounded by $\sum_{p\in\mathcal{P}^\mathcal{L}:f^\mathcal{L}(p)\geq 2} f^\mathcal{L}(p)^2$.

We can efficiently estimate whether there exist ℓ groups in $\mathcal{L}$, each of them containing at least b distinct pairs, by a simple modification of the approach presented in [2]. Building upon work by Bar-Yossef et al. [3], in [2] the authors present an efficient algorithm for estimating the number of distinct pairs in transactional data streams. Essentially, the algorithm hashes each pair in the stream to a random number in (0,1], keeps track of the k smallest hash values and returns as an unbiased estimate of the number of distinct pairs $\lceil k/r_k \rceil$ where r_k is the kth smallest hash value. Using a pairwise independent hash function one shows that we obtain with constant error probability an $(1 \pm \varepsilon)$-approximation of the number of distinct pairs for $k = O(1/\varepsilon^2)$. Running $O(1/\delta)$ copies of the algorithm in parallel and returning the median of the results guarantees an error probability of at most δ. A clever construction of the hash function allows each transaction to be processed in expected linear time.

The above approach admits a natural generalization for estimating the number of pairs occurring exactly i times in $\mathcal{L}$. As shown in [2] the hash function is injective with high probability and thus instead the hash value we can also store the pair with the corresponding hash value. At the end we obtain a sample of pairs with their exact frequency in $\mathcal{L}$ and using standard approaches we estimate the desired quantities.

4 Observations on Datasets

Table 1 summarizes the results for several real datasets. Assuming the transactions are sorted by their length in decreasing order, each row stands for a dataset and the columns give the fraction of pairs generated by the first k percent of the transactions, $1 \leq k \leq 10$. As can be clearly seen from the values in the table, a small ratio of the transactions contributes the majority of pairs. We argue that this is a ubiquitous pattern for many real-life domains.

The first datasets are taken from the FIMI repository. Kosarak contains anonymized click-stream data of a Hungarian on-line news portal, provided by Ferenc Bodon. Webdocs [24] is built from a spidered collection of web html documents. BMS-Web-View1 [18] is built from purchase data of a legwear and legcare web retailer. MovieActors was built from the IMDB database and lists the actors acting in a given movie [7].

The next datasets were built from graphs available at the Stanford Large Network Dataset Collection (http://snap.stanford.edu/data/). For a (directed or undirected) graph given as a set of edges we created a transaction from the neighbors of each vertex. Arxiv COND-MAT [23] (Condense Matter Physics) collaboration network is from the e-print arXiv and covers scientific collaborations between authors papers submitted to Condense Matter category. If an author i co-authored a paper with author j, the graph contains a undirected edge from i to j. Therefore for each author there exists a transaction representing the set of her coauthors. The next two datasets, WikiTalk and WikiVote, are built from the online encyclopedia Wikipedia. For WikiTalk for each Wikipedia user i a transaction records the set of other users j whose talk page was edited at least

Table 1. The ratio of pairs from the top-k percent longest transactions to the total number of pairs for several datasets

Dataset	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Kosarak	0.7834	0.8634	0.9012	0.9237	0.9386	0.9493	0.9572	0.9633	0.9681	0.9719
Webdocs	0.7741	0.8189	0.8467	0.8663	0.8811	0.8930	0.9027	0.9111	0.9182	0.9244
BMS-Web-View1	0.7434	0.8009	0.8347	0.8582	0.8766	0.8920	0.9035	0.9144	0.9224	0.9296
MovieActors	0.8555	0.8991	0.9163	0.9271	0.9351	0.9414	0.9467	0.9512	0.9551	0.9586
CA-CondMat	0.3917	0.5011	0.5709	0.6225	0.6633	0.6965	0.7246	0.7487	0.7695	0.7879
WikiTalk	0.9817	0.9950	0.9979	0.9989	0.9994	0.9996	0.9997	0.9998	0.9998	0.9999
WikiVote	0.5797	0.7179	0.7933	0.8412	0.8749	0.9004	0.9203	0.9352	0.9465	0.9556
web-BerkStan	0.3734	0.5234	0.6005	0.6551	0.6886	0.7179	0.7470	0.7761	0.7997	0.8213
Email-EU-All	0.9895	0.9944	0.9962	0.9973	0.9978	0.9983	0.9986	0.9987	0.9989	0.9991
soc-Epinions1	0.7432	0.8462	0.8977	0.9289	0.9488	0.9619	0.9711	0.9777	0.9825	0.9859

once by i. WikiVote represents popularity of users. For a user i a transaction lists all users j who voted on user i [20,21]. In Web-BerkStan for a page i from the berkely.edu and stanford.edu domains a transaction contains all pages j linked to by i [22]. Email-EU-All is built from email data from a large European research institution. For an email address i a transaction records all email addresses j if i sent at least one message to j [23]. The last dataset, soc-Epinions1, uses data from the general consumer review site Epinions.com. This is a who-trust-whom online social network and members of the site can decide whether to "trust" each other. For a member i a transaction contains all the users trusted by i [29].

5 Experiments

We choose the two datasets Kosarak and Webdocs for our experiments. Kosarak consists of 990,002 transactions over 41,270 items. The total number of pairs is more than 10^8 and the number of distinct pairs is more than 10^7. We took a prefix of Webdocs consisting of the first 100,000 transactions. There are over 5,267,656 items, almost 10^{10} pairs in total and the number of distinct pairs is slightly more than 10^9. Clearly, an exact solution using a hashtable to compute the support of all pairs is infeasible since it will require several Gigabytes of memory. As reported in [6], the distribution of pairs frequencies in both datasets adheres to a power law, therefore an approximation guarantee depending on the residual norm of the frequency vector will yield high quality estimates.

Implementation details. FrequentPairsMiner has been implemented in Java. Experiments have been run on a Mac Pro desktop equipped with Quad-Core Intel 2.8GHz and 8 GB RAM. For Space-Saving we implemented the algorithm as described in [26]. For Frequent we observed that the simpler solution guaranteeing constant amortized cost per update is more efficient than the the linked list data structure presented in [13,17]. We worked with standard Java data structures which required the use of objects. A cache optimized implementation can achieve better space complexity by using only primitive data types. However, the goal of the present paper is to compare our approach with the naïve

application of FREQUENT and SPACE-SAVING to the transactional data stream. Therefore, we don't employ optimization tricks that will equally benefit both approaches.

Parameters. Of crucial importance for the achieved results, both in terms of complexity and accuracy, are the various parameters we set. In particular, how many pairs will the summary record, which transactions will be considered long and how many transactions are we going to keep in the batch. FREQUENT and SPACE-SAVING need a summary of size $O(k/\varepsilon)$ to provide an (ε, k)-guarantee. However, a frequent pair is defined in terms of the number of transactions containing it. Therefore, without prior knowledge on the distribution of transaction lengths it is impossible to predict how long will be the stream of pairs and thus how many pairs we need to record in the summary in order to guarantee that frequent pairs will be in the summary after processing the transactional stream. Jin and Agrawal [16] claim that by adjusting the size of the summary of FREQUENT for each incoming transactions we can guarantee that ε-heavy hitters will be reported but this is erroneous. As a counter example consider a stream of m transactions such that all transactions have length 2. Assume each pair occurs exactly d times in the stream and we have a summary of size $d-1$, such that after processing of the m transactions no pair is recorded in the summary. Since transactions have the same length length we will not update the size of the summary. Then for fixed m, ε and d it is easy to construct a stream of $f(m, \varepsilon, d)$ longer transactions such that some of the pairs become ε-heavy hitters but have exactly the same frequency as pairs that have not appeared among the first m transactions. Thus, the algorithm has no way to distinguish between those pairs. Therefore in the following we assume that the summary size is a user-defined parameter which is to be chosen depending on the available memory. (Note that the parameters k and ε in the theoretical analysis of the algorithm are not user defined but simply used to obtain the best possible bounds on the approximation error.)

From the discussion about the complexity of the algorithm it is clear that a large batch of long transactions will dominate the running time since we need to compute the intersection of each two transactions. Therefore, we implemented the following heuristic: Let b be the size of the summary. In order to achieve good running time, we will keep q long transactions in the batch $\mathcal{L}$, each of them of length at least t, such that $q < t$ and $qt^2 \geq cb$ for some small constant $c > 1$. This will ensure that the running time for finding the intersections of the long transactions is of the order $q^2 \sum t_i$ opposed to $q \sum t_i^2$ for the explicit generation of all pairs in the long transactions. Assuming there are not many intersections among the long transactions, the factor c guarantees that we will (implicitly) find at least b distinct pairs in the batch. If this fails, i.e. it turns out that we have less than b distinct pairs in the batch, we explicitly generate all pairs in the transactions and update the summary with each of them.

Further we observed that the frequent pairs in the batch rarely occur more than a few times. Thus, we simply kept track of a few summary entries with the lowest estimates and when needed, replaced one of them with a heavy entry not recorded in the summary.

Overview of experiments. We compared the straightforward application of FREQUENT and SPACE-SAVING to the stream of pairs to FREQUENTPAIRSMINING with respect to the following criteria:

1. Running time: For different summary sizes we measured the running time.
2. Precision: How many of the reported pairs in the summary are among the top-k pairs for various k.
3. Recall: The ratio of the top-k pairs reported in the summary for various k.
4. Average relative error: Following [10], for the most frequent k pairs we plot $\delta^{rel} = \frac{|f(p)-\widehat{f}(p)|}{f(p)}$, i.e. the absolute difference of the estimate $\widehat{f}(p)$ and the exact count $f(p)$ scaled by $f(p)$.

In all experiments for 2)–4) we set the size of the summary to be 50,000 for kosarak and 100,000 for webdocs. In general, we observe better running times and more precise estimates. The former is a result of the fast processing of long transaction. The latter is due to the fact that in many cases we find a number of distinct pairs which is much bigger than the size of the summary. In order to concentrate on the approximation guarantees achieved by the naïve application of SPACE-SAVING and our optimized approach, we did not implement the book-keeping extension proposed in the original work [26] providing a lower bound on the estimates.

5.1 Experiments for $\mathcal{A}$ =FREQUENT

Running time. As can be seen from Figure 3 we achieve considerably better running time for summary sizes that are not too big. Note that all summary sizes guarantee reasonably good estimates. For larger summaries the advantages of our modified approach become less pronounced since less transactions are considered long.

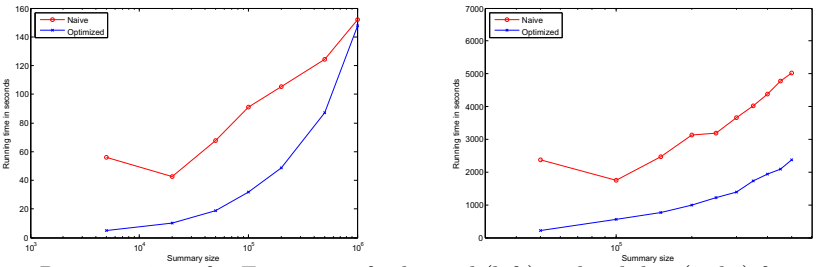


Fig. 3. Running times for FREQUENT for kosarak(left) and webdocs(right) for growing summary size

Precision and recall. In Figures 4 and 5 we plot the precision and recall for the top-k pairs for growing k. Note that the number of pairs explicitly recorded in the summary varies, as we do not explicitly record pairs with a counter set to 0, and for the naïve implementation of FREQUENT it is usually smaller. Nevertheless, FREQUENTPAIRSMINER achieves better precision.

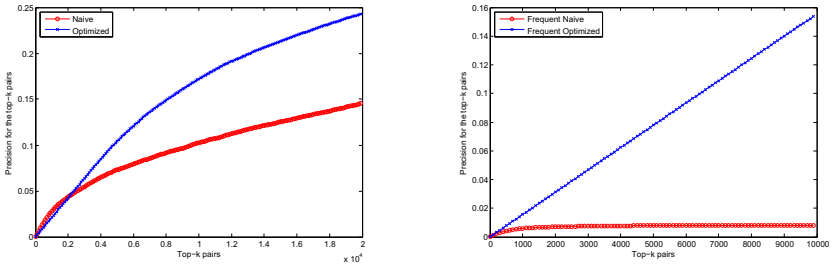


Fig. 4. Precision top-k pairs, k varied from 100 to 20000, for kosarak(left) and webdocs(right)

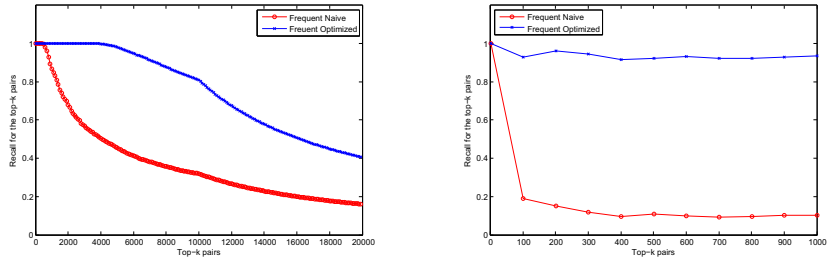


Fig. 5. Recall for FREQUENT for top-k pairs, k varied from 100 to 20000, for kosarak(left) and webdocs(right)

Relative approximation error. The approximation achieved by FREQUENT-PAIRSMINER is also considerably better which can be explained with the smaller number of pair weights decrements.

5.2 Experiments for $\mathcal{A}$ =SPACE-SAVING

Running time. The running time measurements are presented in Figure 7. The larger difference compared to FREQUENT seems to be due to the fact that the summary size for SPACE-SAVING remains constant.

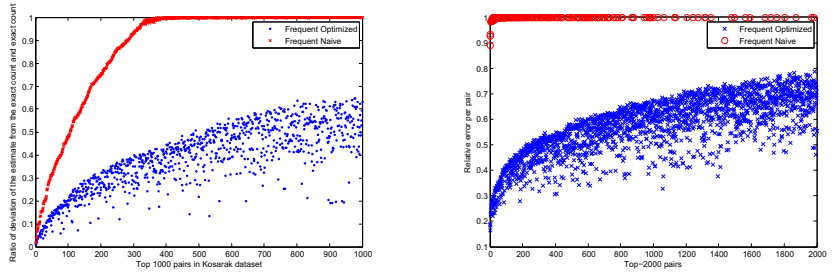


Fig. 6. Average relative error for FREQUENT for kosarak(left) and webdocs(right)

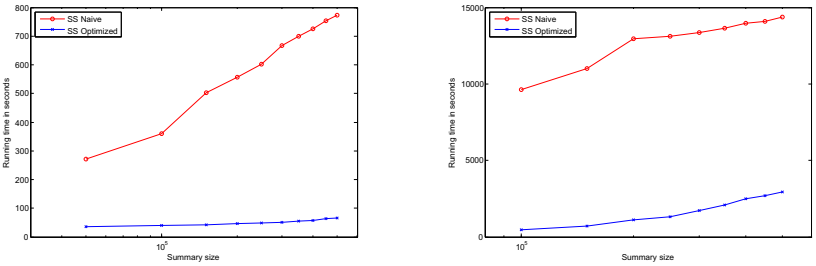


Fig. 7. Running times for SPACE-SAVING for kosarak(left) and webdocs(right)

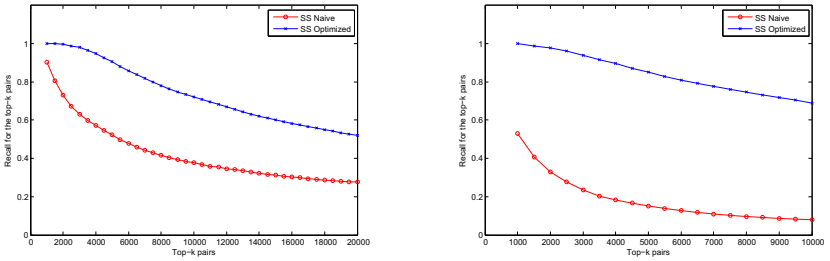


Fig. 8. Recall for Space-Saving for kosarak(left) and webdocs(right)

Recall. The number of returned pairs in SPACE-SAVING always equals the size of the summary, thus results on precision will be redundant from results on recall. Figure 8 shows the recall for the two considered datasets.

Relative approximation error. Figure 9 presents results for the relative approximation error. The straight line parallel to the x-axis is from pairs that have not been reported.

6 Further Directions

We presented evidence that meanwhile classic frequent items mining streaming algorithms can be considerably improved when applied to transactional streams

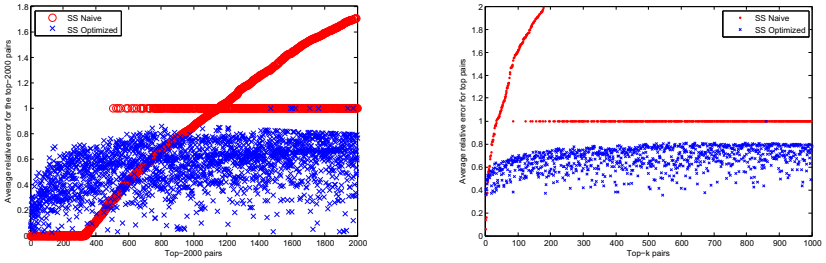


Fig. 9. Average relative error for Space-Saving for kosarak(left) and webdocs(right)

when the transactions lengths are skewed. A more thorough research is needed to fully employ the benefits of our approach. In particular, other approaches for counting exactly frequent pairs in a few long transaction should be possible. Note that the current approach relies on the observation on real datasets that the overlap in the long transactions in the batch is small. However, for other datasets this assumption might not be feasible and the intersections approach will produce poor results. In such a case applying a standard approach like Apriori to the transactions in the batch might be more suitable.

Another direction is to apply our technique to mining frequent items from transactional data streams over sliding windows. In the sliding window model, we are interested in pairs occurring a certain number of times in the last t transactions for a user-defined t. A natural candidate to apply (a modification of) our approach is the algorithm by Lee and Ting [19]. This algorithm builds upon FREQUENT and detects frequent items among the most recently seen t items.

Acknowledgements. I would like to thank my supervisor Rasmus Pagh for helpful discussions.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: VLDB 1994, pp. 487–499 (1994)
2. Amossen, R.R., Campagna, A., Pagh, R.: Better Size Estimation for Sparse Matrix Products. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX 2010, LNCS, vol. 6302, pp. 406–419. Springer, Heidelberg (2010)
3. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D., Trevisan, L.: Counting Distinct Elements in a Data Stream. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 1–10. Springer, Heidelberg (2002)
4. Berinde, R., Indyk, P., Cormode, G., Strauss, M.J.: Space-optimal heavy hitters with strong error bounds. ACM Trans. Database Syst. 35(4), 26 (2010)
5. Bose, P., Kranakis, E., Morin, P., Tang, Y.: Bounds for Frequency Estimation of Packet Streams. In: SIROCCO 2003, pp. 33–42 (2003)
6. Campagna, A., Kutzkov, K., Pagh, R.: Frequent Pairs in Data Streams: Exploiting Parallelism and Skew. In: ICDM Workshops 2011, pp. 145–150 (2011)
7. Campagna, A., Pagh, R.: Finding Associations and Computing Similarity via Biased Pair Sampling. In: ICDM 2009, pp. 61–70 (2009)
8. Campagna, A., Pagh, R.: On Finding Similar Items in a Stream of Transactions. In: ICDM Workshops 2010, pp. 121–128 (2010)
9. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. Theor. Comput. Sci 312(1), 3–15 (2004)
10. Cormode, G., Hadjieleftheriou, M.: Finding the frequent items in streams of data. ACM Commun. 52(10), 97–105 (2009)
11. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. J. Algorithms 55(1), 58–75 (2005)
12. Cormode, G., Muthukrishnan, S.: Summarizing and Mining Skewed Data Streams. In: SDM 2005 (2005)
13. 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)

14. Han, J., Pei, J., Yin, Y., Mao, R.: Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Min. Knowl. Discov. 8(1), 53–87 (2004)
15. Jiang, N., Gruenwald, L.: Research issues in data stream association rule mining. SIGMOD Record 35(1), 14–19 (2006)
16. Jin, R., Agrawal, G.: An Algorithm for In-Core Frequent Itemset Mining on Streaming Data. In: ICDM 2005, 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, 51–55 (2003)
18. Kohavi, R., Brodley, C.E., Frasca, B., Mason, L., Zheng, Z.: KDD-Cup 2000 Organizers' Report: Peeling the Onion. SIGKDD Explorations 2(2), 86–98 (2000)
19. Lee, L.K., Ting, H.F.: A simpler and more efficient deterministic scheme for finding frequent items over sliding windows. In: PODS 2006, pp. 290–297 (2006)
20. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Signed Networks in Social Media. In: CHI 2010 (2010)
21. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Predicting Positive and Negative Links in Online Social Networks. In: WWW 2010 (2010)
22. Leskovec, J., Lang, K., Dasgupta, A., Mahoney, M.: Community Structure in Large Networks. Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Mathematics 6(1), 29–123 (2009)
23. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph Evolution. Densification and Shrinking Diameters. ACM TKDD 1(1) (2007)
24. Lucchese, C., Orlando, S., Perego, R., Silvestri, F.: WebDocs: a real-life huge transactional dataset. In: FIMI 2004 (2004)
25. Manku, G.S., Motwani, R.: Approximate Frequency Counts over Data Streams. In: VLDB 2002, pp. 346–357 (2007)
26. Metwally, A., Agrawal, D., El Abbadi, A.: An integrated efficient solution for computing frequent and top-k elements in data streams. ACM Trans. Database Syst. 31(3), 1095–1133 (2006)
27. Misra, J., Gries, D.: Finding Repeated Elements. Sci. Comput. Program. 2(2), 143–152 (1982)
28. Park, J.S., Chen, M.-S., Yu, P.S.: Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. IEEE TKDE 9(5), 813–825 (1997)
29. 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)
30. Yu, J.X., Chong, Z., Lu, H., Zhang, Z., Zhou, A.: A false negative approach to mining frequent itemsets from high speed transactional data streams. Inf. Sci. 176(14), 1986–2015 (2006)

Mirror Descent for Metric Learning: A Unified Approach

Gautam Kunapuli and Jude Shavlik

University of Wisconsin-Madison, USA

Abstract. Most metric learning methods are characterized by diverse loss functions and projection methods, which naturally begs the question: is there a wider framework that can generalize many of these methods? In addition, ever persistent issues are those of scalability to large data sets and the question of kernelizability. We propose a unified approach to Mahalanobis metric learning: an online regularized metric learning algorithm based on the ideas of composite objective mirror descent (COMID). The metric learning problem is formulated as a regularized positive semi-definite matrix learning problem, whose update rules can be derived using the COMID framework. This approach aims to be scalable, kernelizable, and admissible to many different types of Bregman and loss functions, which allows for the tailoring of several different classes of algorithms. The most novel contribution is the use of the trace norm, which yields a sparse metric in its eigenspectrum, thus simultaneously performing feature selection along with metric learning.

1 Introduction

The concept of similarity, or metric, is central to many well-known algorithms such as k-means clustering [1], k-nearest neighbors [2], multi-dimensional scaling [3] and semi-supervised clustering [4]. While there are many approaches to metric learning, a large body of work is focussed on learning the Mahalanobis distance, which amounts to learning a feature-space transformation and computing the distance in the transformed space. Among these approaches are the work of Xing et al., [5], the large-margin nearest neighbor (LMNN) algorithm [6], information-theoretic metric learning (ITML) [7] and BoostMetric [8]. In addition to the batch approaches above, online algorithms such the pseudo-metric online learning algorithm (POLA) [9] have also been developed. These approaches have been applied successfully to a diverse range of real-world applications such as face verification [10] and road-lane detection [4].

The goal of a metric learning approach is to learn a *distance function*, typically from additional information about the data set. In the supervised and semi-supervised classification setting, the notion of similarity or dissimilarity can be inferred from the class information available from the labels. Thus, if two data points are in the same class, they are assumed to be similar to each other, while two points in different classes are assumed to be dissimilar. The learned metric should ensure that distance between dissimilar points is larger than distance

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 859–874, 2012.

between similar points. Such a metric, can then be used different semi-supervised and unsupervised learning methods such as k-means clustering.

In this work, we consider the Mahalanobis metric learning problem applied to k-nearest neighbors classification. The Mahalanobis metric is a distance function we learn that is of the form $d(\mathbf{x}, \mathbf{z}) = \|L\mathbf{x} - L\mathbf{z}\|_2$. Thus, we hope to learn a transformation of the data L that separates dissimilar points and brings similar points closer, and we measure distance in this transformed space.

For the remainder of this section, we discuss the problem setting and in Section 2, we introduce composite mirror descent for metric learning. We derive the general update rules, and discuss their implementation details from the perspective of efficiency in Sections 3 and 4. The kernel version of this approach is introduced in Section 5. This method is closely related to several other well-known metric learning approaches and this aspect is discussed in Section 6. In Section 7, we compare the mirror descent approach with some well-known metric learning methods on different data sets, and conclude in Section 8.

1.1 Problem Setting

We wish to learn a Mahalanobis metric $d(\mathbf{x}, \mathbf{z})$ over a feature space $\mathcal{X} \subseteq \mathbb{R}^n$. The metric is a distance function that is used to measure similarity between two instances $\mathbf{x}$ and $\mathbf{z}$ in feature space and satisfies three conditions: $d(\mathbf{x}, \mathbf{z}) \geq 0$ (non-negativity), $d(\mathbf{x}, \mathbf{z}) = d(\mathbf{z}, \mathbf{x})$ (symmetry), and $d(\mathbf{x}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{w}) + d(\mathbf{w}, \mathbf{z})$ (sub-additivity). We formulate the problem in the spirit of Shalev-Shwartz et al., [9], where the goal is to incrementally learn a metric, given triplets of the form $(\mathbf{x}_t, \mathbf{z}_t, y_t)_{t=1}^T$. The label $y_t = 1$ indicates that training point $\mathbf{x}_t$ is similar to $\mathbf{z}_t$ and $y = -1$ indicates dissimilarity.

The metric we learn is of the form $d(\mathbf{x}, \mathbf{z}) = \|L(\mathbf{x} - \mathbf{z})\|_2$, where $L \in \mathbb{R}^{n \times n}$ is a linear transformation. Since learning this metric directly is difficult owing to non-convexity, we consider instead:

$$d_M(\mathbf{x}, \mathbf{z})^2 = (\mathbf{x} - \mathbf{z})' L' L (\mathbf{x} - \mathbf{z}) = (\mathbf{x} - \mathbf{z})' M (\mathbf{x} - \mathbf{z}), \tag{1}$$

with $M \in \mathbb{S}_+^n$, the cone of positive semi-definite (psd) matrices. Given T labeled pairs of points $(\mathbf{x}_t, \mathbf{z}_t, y_t)_{t=1}^T$, we learn (M, μ) such that similar points are transformed to be closer to each other, which dissimilar points are transformed to be farther from each other. This condition can be formulated via the constraints

$$\begin{aligned} \forall (\mathbf{x}, \mathbf{z}, y = +1) \;&\Rightarrow\; d_M(\mathbf{x}, \mathbf{z})^2 \leq \mu - 1, \\ \forall (\mathbf{x}, \mathbf{z}, y = -1) \;&\Rightarrow\; d_M(\mathbf{x}, \mathbf{z})^2 \geq \mu + 1, \end{aligned} \tag{2}$$

which can be written simply as $y_t(\mu - d_M(\mathbf{x}_t, \mathbf{z}_t)^2) \geq 1$. Note that we cannot have $\mu < 1$ as it implies via the constraints (2) that the distance is negative. We define the margin function for a pair of instances $\mathbf{x}_t$ and $\mathbf{z}_t$, given a label y_t, as

$$m(\mathbf{x}_t, \mathbf{z}_t, y_t) = y_t(\mu - d_M(\mathbf{x}_t, \mathbf{z}_t)^2) = y_t \left(\mu - (\mathbf{x}_t - \mathbf{z}_t)' M (\mathbf{x}_t - \mathbf{z}_t)\right). \tag{3}$$

This lets us define several loss functions, for instance, the hinge-loss: $\ell_t(M, \mu)$ $= \max\{\, 0,\, 1 - m(\mathbf{x}_t, \mathbf{z}_t, y_t) \,\}$. The behavior of such loss functions can be observed

in the one-dimensional example in Figure 1. When points z near $x = -0.5$ are labeled similar (Figure 1, left), and their distance measured through the metric M is under the threshold μ, the loss is zero or small. Similarly labeled points z that are far away from $x = -0.5$ are penalized, with the penalty increasing with the distance. In contrast, dissimilarly labeled points near $x = -0.5$ suffer a high loss (Figure 1, right), while those that are sufficiently far away according to the threshold μ are not penalized. It should be noted that μ controls the width of sensitivity around x. Loss functions are discussed further in Section 2.1.

In addition to learning a metric that minimizes this notion of loss, we also incorporate regularization into the problem so that the resulting metric has sparsity. It is well-known that ℓ_1-regularization yields sparse solutions [11]; analogously, minimizing the trace-norm of M i.e., the sum of the singular values of M yields sparsity in the spectrum of M, thus minimizing the rank of M [12]. Given T samples, the overall problem is one of regularized loss minimization, which leads to an optimization problem of the form

$$\min_{M \succeq 0, \mu \geq 1} \frac{1}{T} \sum_{t=1}^{T} \ell_t(M, \mu) + \rho\, r(M), \tag{4}$$

where the loss function $\ell_t : \mathbb{S}_+^n \times \mathbb{R} \to \mathbb{R}$ and the regularization function $r : \mathbb{S}_+^n \to \mathbb{R}$ are both convex and $\rho \in \mathbb{R}_+$ is the regularization parameter. Note that the minimization step in (4) contains a matrix projection into $\mathbb{S}_+^n$, which is a consequence of constraining $M \succeq 0$, and a scalar projection of $\mu \geq 1$. Before describing the proposed approach, we introduce some notation.

1.2 Notation and Background

Scalars are denoted in lower-case (μ), vectors in bold face ($\mathbf{v}$), and matrices in upper case (M). The vector $\mathbf{e}$ denotes a vector of ones, while I denotes the identity matrix, with the dimension of either being apparent from the context. For a vector $\mathbf{v}$, the plus function $\mathbf{v}_+$ is defined as the componentwise maximum with respect to zero i.e., $(v_j)_+ = \max\{0, v_j\}$, for the j-th component of $\mathbf{v}$. The

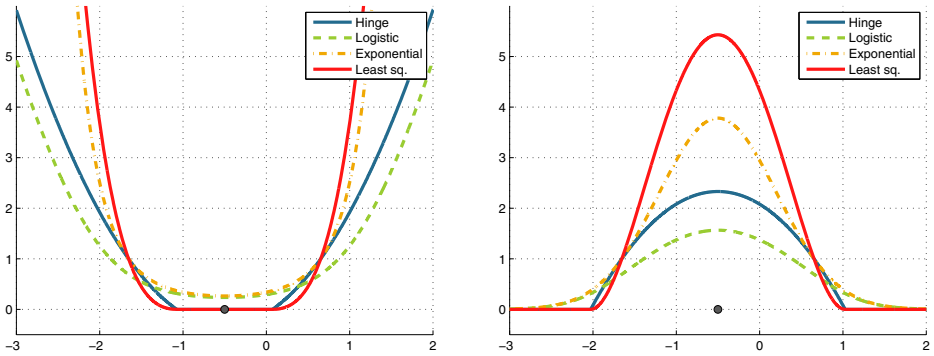


Fig. 1. Behavior of different loss functions: (**left**) $y = +1$; (**right**) $y = -1$

step function $\mathbf{v}_\star$ is defined componentwise as $(v_j)_\star = v_j$ if $v_j > 0$ and 0 if $v_j \leq 0$. The inner product of matrices X and Y is defined as using the trace: $\langle X, Y \rangle = \mathsf{tr}\, X'Y$. We can also compute matrix gradients; one particularly useful definition is $\nabla_X \langle X, Y \rangle = Y$. We write the singular value decomposition of $X = U\Sigma V$, while for symmetric matrices, we can write $X = V\Lambda V'$. Matrix functions $f(X)$ (e.g., $\exp X$, $\log X$) can be computed via the eigen-decomposition of X, that is, $f(X) = Vf(\Lambda)V'$. The Frobenius norm of X is denoted $\|X\|_F = \sqrt{\langle X, X \rangle}$; the trace norm of X is denoted $|||X||| = \mathbf{e}'\boldsymbol{\sigma}$, where $\boldsymbol{\sigma}$ are the singular values of X. For symmetric matrices, $|||X||| = \mathbf{e}'|\boldsymbol{\lambda}|$, where $\boldsymbol{\lambda}$ are the eigenvalues of X.

The Bregman divergence [13] with respect to a strictly convex function ψ is defined as $B_\psi(\mathbf{x}, \mathbf{z}) = \psi(\mathbf{x}) - \psi(\mathbf{z}) - \nabla\psi(\mathbf{z})'(\mathbf{x} - \mathbf{z})$. For example, the function $\psi(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2$ is a Bregman function which induces the squared-Euclidean norm, $B_\psi(\mathbf{x}, \mathbf{z}) = \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2$. This definition can naturally be extended to convex functions defined over matrices i.e., $B_\psi(X, Z) = \psi(X) - \psi(Z) - \langle \nabla\psi(Z), (X - Z) \rangle$. A well-known example is the squared-Frobenius norm $B_\psi(X, Z) = \frac{1}{2}\|X - Z\|_F^2$, induced by $\psi(X) = \frac{1}{2}\|X\|_F^2$.

2 Mirror Descent for Metric Learning

The mirror descent algorithm [14] is an iterative proximal-gradient method for minimizing a convex function, $\phi : \Omega \to \mathbb{R}$. Based on this approach, an update in the online setting, with function ϕ_t is

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \Omega} B_\psi(\mathbf{w}, \mathbf{w}_t) + \eta\, \nabla'\phi_t(\mathbf{w}_t)(\mathbf{w} - \mathbf{w}_t). \tag{5}$$

Recently, Duchi et al., [15] generalized mirror descent to the case where the functions $\phi_t = \ell_t + r$ are composite, consisting of loss and regularization terms:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \Omega} B_\psi(\mathbf{w}, \mathbf{w}_t) + \eta\, \nabla'\ell_t(\mathbf{w}_t)(\mathbf{w} - \mathbf{w}_t) + \eta\, r(\mathbf{w}). \tag{6}$$

The subtle, yet significant difference between (5) and (6) is that the entire composite function ϕ_t is not linearized. Rather, only ℓ_t is linearized; this leads to the composite mirror descent algorithm (COMID). The reason for partial linearization is because general mirror descent applied to ℓ_1-regularization does not lead to sparse updates, whereas the COMID update does.

We utilize the framework of Duchi et al., to formulate metric learning as an online problem. However, we compute matrix update rules directly rather than derive passive-aggressive-like online updates of [9]. The goal is to optimize the objective (4) in an online manner i.e., at each iteration $t = 1, \ldots, T$, the algorithm receives a labeled pair of points $(\mathbf{x}_t, \mathbf{z}_t, y_t)$, which has an associated loss function $\ell_t(M, \mu)$, and the estimates M_{t+1} and μ_{t+1} are calculated using a composite mirror descent update rule. Since we are interested in sparse updates as well, we use the trace norm, $|||M|||$, the effect of which is controlled via a regularization parameter $\rho > 0$. We derive generalized update rules for a general

loss function and Bregman divergence. At each step, we compute updates given a learning rate $\eta > 0$, and regularization parameter $\rho > 0$ according to

$$M_{t+1} = \arg\min_{M \succeq 0} B_\psi(M, M_t) + \eta \langle \nabla_M \ell_t(M_t, \mu_t), M - M_t \rangle + \eta \rho \, |||M|||, \quad (7)$$

$$\mu_{t+1} = \arg\min_{\mu \geq 1} \quad B_\psi(\mu, \mu_t) + \eta \nabla_\mu \ell_t(M_t, \mu_t)' (\mu - \mu_t). \quad (8)$$

This metric learning formulation[1] has several advantages:

1. **General framework**. Different classes of algorithms can be designed based on different Bregman and loss functions. For example, using the Euclidean distance as the Bregman function produces additive updates, while using relative entropy produces multiplicative updates.
2. **Scalable to large data sets**. As we show below, the matrix updates involve the computation of a rank-1 update to the current eigendecomposition of M, which is highly efficient. Furthermore, the rank-one eigen-update scheme discussed here is embarrassingly parallelizable.
3. **Trace-norm regularization produces sparse metric**. The trace-norm regularization ensures that at each step, the updated $M = L'L$ is sparse in its spectrum of singular/eigenvalues. The trace-norm, like the ℓ_1 norm, introduces sparsity into the eigenvalues of M. The solution typically has $r < n$ non-zero eigenvalues: $\tilde{L} = V_r \Lambda_r V_r'$ and the approach performs input-space feature selection.
4. **Theoretical regret guarantees**. In the online optimization setting, for a strongly-convex Bregman function B_ψ and a Lipschitz continuous loss function ℓ_t, COMID has $O(\sqrt{T})$ regret [15]. Furthermore, strong convexity of the composite function ensures $O(\log T)$ regret.
5. **Kernelizable for nonlinear metric learning**. Many existing distance learning methods are not intuitively kernelizable. Recently, Chatpatanasiri et al., [16] showed various techniques of kernelizing some popular metric learning approaches. Their results are easily extended to this approach in order to learn nonlinear metrics.

2.1 Loss Functions

Recall that the margin for a labeled pair of points $(\mathbf{x}_t, \mathbf{z}_t, y_t)$ is defined as $m_t(\mathbf{u}_t, y_t) = y_t(\mu - \mathsf{tr}\, M \mathbf{u}_t \mathbf{u}_t')$, with $\mathbf{u}_t = \mathbf{x}_t - \mathbf{z}_t$. When a pair $(\mathbf{x}_t, \mathbf{z}_t)$ are similar with $y_t = 1$, any loss function should produce zero (or small loss) if the distance according to the learned metric is less than a threshold i.e., $\mathsf{tr}\, M \mathbf{u}_t \mathbf{u}_t' \leq \mu$. For dissimilar points ($y_t = -1$), when $\mathsf{tr}\, M \mathbf{u}_t \mathbf{u}_t' \leq \mu$, the loss suffered is higher. Many such loss functions can be used with the update rules (7-8). Table 1 and Figure 1 show some common loss functions. It is interesting to note that all the loss functions in Table 1 have gradients of the form $\alpha \mathbf{u}\mathbf{u}'$. As we show in Section 4, we can exploit this fact to implement update rules more efficiently.

[1] As the trace-norm allows us to learn a low-rank matrix M, this problem is an instance of *pseudo-metric* learning; directions in the null space of M cannot be differentiated.

Table 1. Examples of some loss functions, and their gradients with respect to M and μ. For a pair of instances, $\mathbf{u}_t = \mathbf{x}_t - \mathbf{z}_t$, and $m_t(\mathbf{u}_t; M, \mu) = y_t(\mu - \mathsf{tr}\, M\mathbf{u}_t\mathbf{u}_t')$.

Loss	$\ell_t(M_t,\, \mu_t)$	$\nabla_M\, \ell_t(M_t,\, \mu_t)$	$\nabla_\mu\, \ell_t(M_t,\, \mu_t)$
Hinge	$(1 - m_t)_+$	$(1 - m_t)_\star\ (y_t\mathbf{u}_t\mathbf{u}_t')$	$-\,(1 - m_t)_\star\, y_t$
Modified Least Sq.	$\frac{1}{2}\,(1 - m_t)_+^2$	$(1 - m_t)_+\,(y_t\mathbf{u}_t\mathbf{u}_t')$	$-(1 - m_t)_+ y_t$
Exponential	$\exp(-m_t)$	$\exp(-m_t)\,(y_t\mathbf{u}_t\mathbf{u}_t')$	$-\exp(-m_t) y_t$
Logistic	$\log\,(1 + \exp(-m_t))$	$\frac{\exp(-m_t)}{1+\exp(-m_t)}\,(y_t\mathbf{u}_t\mathbf{u}_t')$	$-\frac{\exp(-m_t)}{1+\exp(-m_t)} y_t$

2.2 Bregman Divergences

Bregman divergences have been extensively studied in literature; see, for instance, Censor and Zenios [17]. The strong convexity of Schatten and entropic matrix functions, which we use here, was recently established by Kakade et al., [18]. Slightly abusing notation, we use $B_\psi(\cdot,\, \cdot)$ for both scalars and matrices.

The squared p-norms $\psi(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_p^2$ are strongly convex and induce Bregman divergences. For a matrix X, the definition can be extended to Schatten p-norms [19], a family of unitary norms defined by applying the p-norm to its singular values: $\psi(X) = \frac{1}{2}\|\boldsymbol{\sigma}(X)\|_p^2$. With $p = 2$, we obtain the *squared-Euclidean distance* $B_\psi(\mathbf{x}, \mathbf{z}) = \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2$ for scalars, and the *squared-Frobenius distance* i.e., $B_\psi(X, Z) = \frac{1}{2}\|X - Z\|_F^2$ for matrices.

The function $\psi(\mathbf{x}) = \sum_i x_i \log x_i - x_i$ induces the *Kullback-Liebler (KL) divergence*, $B_\psi(\mathbf{x}, \mathbf{z}) = \sum_i x_i \log \frac{x_i}{z_i} - x_i + z_i$. For a matrix X, if λ_i is the i-th eigenvalue, the Bregman function can be analogously extended: $\psi(X) = \sum_i \lambda_i \log \lambda_i - \lambda_i = \mathsf{tr}\, X \log X - X$ giving us the *von Neumann divergence*, $B_\psi(X, Y) = \mathsf{tr}\,(X \log X - X \log Y - X + Y)$.

3 Deriving Update Rules for M_{t+1} and μ_{t+1}

The update rule (7) can be broken down into two separate updates:

$$M_{t+\frac{1}{2}} = \arg\min_M\ B_\psi(M,\, M_t) \,+\, \eta\,\langle\, \nabla_M \ell_t(M_t, \mu_t),\, M - M_t\,\rangle, \tag{9}$$

$$M_{t+1} = \arg\min_{M \succeq 0}\ B_\psi(M,\, M_{t+\frac{1}{2}}) \,+\, \eta\rho\ |||\, M\, |||\,. \tag{10}$$

The gradient condition of (9): $0 \in \nabla\psi(M_{t+\frac{1}{2}}) - \nabla\psi(M_t) + \eta\,\nabla_M \ell_t(M_t, \mu_t)$, gives us the intermediate solution: $M_{t+\frac{1}{2}} = \nabla\psi^{-1}\,(\,\nabla\psi(M_t) - \eta\,\nabla_M \ell_t(M_t, \mu_t)\,)$, which can be used to solve (10). The latter is closely related to the trace-norm minimization problem, which was solved by Cai et al., [20]:

$$\underset{X}{\text{minimize}}\ \ B_\psi(X, Y) \,+\, \tau\ |||\, X\, |||\,. \tag{11}$$

Cai et al., showed that when $\psi(X) = \frac{1}{2}\,\|X\|_F^2$, the optimal solution to (11) is $\Sigma_\tau(Y)$, where Σ_τ is the *singular-value thresholding/shrinkage operator*. For $X \in \mathbb{R}^{m\times n}$, with SVD $X = U\,\mathsf{diag}(\boldsymbol{\sigma})\,V'$, the singular-value shrinkage operator

is defined as $\Sigma_\tau(X) = U\,\mathsf{diag}(\boldsymbol{\sigma}_\tau)\,V'$, where $(\sigma_\tau)_i = (\sigma_i - \tau)_+$. Thus, Σ_τ shrinks all singular values by $\tau > 0$, and cuts off those below the specified threshold to zero i.e., those $\sigma_i \leq \tau$. This problem (10) differs from (11) in two key ways:

- The solution is derived for $X, Y \in \mathbb{R}^{m\times n}$, and assumes unitarily invariant Bregman functions $\psi(X)$. It relies on an elegant result by Lewis [21] that shows that for a *unitarily invariant* matrix function $\psi(X)$ (i.e., $\psi(PXQ) = \psi(X)$, for any P, Q unitary), the subdifferential can be calculated as $\nabla\psi(X) = U\,\mathsf{diag}(\nabla\psi(\boldsymbol{\sigma}))\,V'$. However, all Bregman functions are *not* unitarily invariant[2], and consequently, it is not possible to characterize the subgradients in our general case. Fortunately, we are interested in symmetric $X \in \mathbb{S}^n_+$, and in these cases, an analogous result by Lewis [22] characterizes subgradients of *spectral functions* $\psi(X)$ as $\nabla\psi(X) = V\,\mathsf{diag}(\nabla\psi(\boldsymbol{\lambda}))\,V'$, given the eigendecomposition $X = V\mathsf{diag}(\boldsymbol{\lambda})V'$. The symmetry of X ensures that $\psi(X)$ are *orthogonally invariant* (i.e., $\psi(QXQ') = \psi(X)$, for any orthogonal Q).
- No positivity constraints $X \succeq 0$ are imposed in (11). In our case, since X is a pseudo-metric, we need to ensure that it is positive semidefinite. As we show below, we can derive a closed-form solution, taking into account that $X \succeq 0$, using the *modified eigenvalue thresholding operator*, $S_\tau(X) = V\,\mathsf{diag}(\boldsymbol{\lambda}_\tau)\,V'$, where $(\lambda_\tau)_i = (\lambda_i - \tau)_+$, ensuring that *all eigenvalues below the threshold* τ *are cut off*, including all negative eigenvalues.

Proposition 1. *The optimal solution to (10) is given by*

$$M_{t+1} \;=\; \nabla\psi^{-1}\Big(S_{\eta\rho}(\nabla\psi(M_{t+\frac{1}{2}}))\Big) \;=\; V\,\nabla\psi^{-1}\left(\,\mathit{diag}(S_{\eta\rho}(\boldsymbol{\lambda}))\,\right)V', \tag{12}$$

where $\nabla\psi(M_{t+\frac{1}{2}}) \;=\; \nabla\psi(M_t) - \eta\,\nabla_M\ell_t(M_t,\mu_t) \;=\; V\,\mathit{diag}(\boldsymbol{\lambda})\,V'$.

Note here that the computation of $\nabla\psi(M_{t+\frac{1}{2}})$ involves a symmetric rank-one update because the gradient of the loss function $\nabla_M\ell_t$ is a rank-one matrix (see Table 1). The update essentially consists of computing a symmetric rank-one update to the current eigendecomposition of the pseudo-metric and then cutting off all eigenvalues $< \eta\rho$. We discuss this step further in Section 4. Finally, a closed-form update can be derived for μ_{t+1} as well, from the formulation (8). In this case, the intermediate solution $\mu_{t+\frac{1}{2}}$ is projected onto $\mu \geq 1$.

Proposition 2. *The optimal solution to (8) is given by*

$$\mu_{t+1} = \max\left(\nabla\psi^{-1}\left(\nabla\psi(\mu_t) - \eta\,\nabla_\mu\ell_t(M_t,\mu_t)\right),\;\; 1\right). \tag{13}$$

4 Implementing Update Rules for M_{t+1}

At the t-th iteration, with $M_t = V_t\nabla\psi(\Lambda_t)V_t'$, we have:

$$\begin{aligned}
\text{(Intermediate gradient)} \qquad \nabla\psi(M_{t+\frac{1}{2}}) \;&=\; V_t\nabla\psi(\Lambda_t)V_t' - \alpha\mathbf{u}_t\mathbf{u}_t'\\
\text{(EVD of intermediate gradient)}\quad \nabla\psi(M_{t+\frac{1}{2}}) \;&=\; V_{t+1}\,\Lambda_{t+1}\,V_{t+1}'\\
\text{(Matrix update/thresholding)}\quad M_{t+1} \;&=\; V_{t+1}\,\nabla\psi^{-1}\left(\,S_{\eta\rho}(\Lambda_{t+1})\,\right)V_{t+1}'
\end{aligned}$$

[2] An example is the entropy function that induces the von Neumann divergence: rotations applied by arbitrary matrices P, Q could change the sign of the eigenvalues.

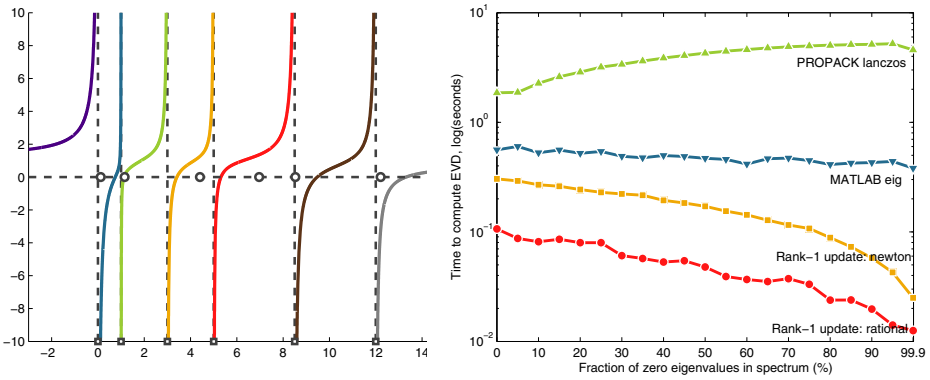


Fig. 2. (**left**) The interleaving of eigenvalues λ_i ($\square$) of a matrix $M \in \mathbb{S}^6$ with the eigenvalues μ_i ($\circ$) of a rank-one perturbation $\tilde{M} = M + \rho \mathbf{u}\mathbf{u}'$; (**right**) Comparing the efficiency of Newton's method and the rational interpolation method with Lanczos and `MATLAB`'s `eig` which uses QR + Householder. The approaches are compared on 20 random matrices in $\mathbb{S}^{500}_{+}$, over increasing sparsity in the spectrum of the matrix.

From Table 1, we observe that the gradient of the loss function is generally of the form $\ell_t = \alpha \mathbf{u}_t \mathbf{u}_t'$, where $\mathbf{u}_t = \mathbf{x}_t - \mathbf{z}_t$. The first two steps constitute a *rank-one update* of the eigendecomposition at the current iteration, which is the most expensive step. While there are several well-known approaches (power iteration, Lanczos, QR + Householder; see [23]), we discuss a different approach that exploits the the *eigenvalue interlacing property* [23, Chapter 8] (Figure 2, left) to significantly improve efficiency.

If $M_t = V\, \mathsf{diag}(\boldsymbol{\lambda})\, V'$, with eigenvalues $\lambda_1 \leq \ldots \leq \lambda_n$, then a symmetric rank-one update is $M_{t+1} = M_t + \alpha \mathbf{u}\mathbf{u}'$, with $M_{t+1} = W\, \mathsf{diag}(\boldsymbol{\mu})\, W'$, and eigenvalues $\mu_1 \leq \ldots \leq \mu_n$. The eigenvalues are related by the secular equation

$$f(\mu) \;:=\; 1 - \alpha \mathbf{u}'(\mu I_n - \mathsf{diag}(\boldsymbol{\lambda}))^{-1}\mathbf{u} \;=\; 0,$$

and the eigenvalues *interlace* i.e., if $\alpha > 0$, $\lambda_1 \leq \mu_1 \leq \lambda_2 \leq \mu_2 \ldots \leq \lambda_n \leq \mu_n$; and if $\alpha < 0$, $\mu_1 \leq \lambda_1 \leq \mu_2 \leq \lambda_2 \leq \ldots \leq \mu_n \leq \lambda_n$. The eigenvectors can also be easily updated as

$$\mathbf{w}_i = \mathbf{v}_i(\mu I_n - \mathsf{diag}(\boldsymbol{\lambda}))^{-1}\mathbf{u},$$

where $\mathbf{w}_i$ and $\mathbf{v}_i$ are the i-th columns of W and V respectively. Now, since we know that the i-th eigenvalue of M_{t+1}, $\mu_i \in (\lambda_i, \lambda_{i+1})$, for $\alpha > 0$ (and $\mu_i \in (\lambda_{i-1}, \lambda_i)$, for $\alpha < 0$) any root-finding method such as Newton-Raphson safeguarded with bisection search can be used to find μ_i. Another consequence of interlacing is that if $\lambda_i = \lambda_{i+1} = \ldots = \lambda_{i+k} = \lambda$, i.e., there are k repeated eigenvalues, then we can avoid the computation of $k-1$ eigenvalues (and eigenvectors) in the update since $\mu_i = \mu_{i+1} = \ldots = \mu_{i+k-1} = \lambda$ as well. This is particularly suitable for our purposes since we seek to introduce more zero eigenvalues into the spectrum of the metric. We discuss some specific details of our implementation:

- **Numerical stability.** General root-finding approaches introduce numerical instability, which propagates into the computation of the eigenvectors leading to non-orthogonality. This issue is addressed by the rational interpolation approach of Gu and Eisenstat [24], which we implement. The approach is based on the observation that while Newton's method uses a local linear approximation of the secular equation, since the secular equation is rational, better stability can be obtained through a local rational approximation.
- **Learning rate.** We use an adaptive learning rate, $\eta_t = \eta/\sqrt{t}$, which gives $O(\sqrt{T})$ regret. The approach requires the user to select the learning rate η and the parameter ρ, which controls the sparsity of the learned metric.
- **Low Rank Learning.** As, the von Neumann divergence is undefined for low-rank matrices, we compute updates using the *reduced eigendecomposition*, $M_t = \tilde{V}_t \tilde{\Lambda}_t \tilde{V}_t'$, where $\tilde{V}_t$ and $\tilde{\Lambda}_t$ correspond only to the r non-zero eigenvalues. This is similar to the approach of Kulis et al., [25] for low-rank kernel learning. As a result of applying $\nabla\psi^{-1} = \exp(\cdot)$ to the updated eigenvalues in (12), the smallest eigenvalues in the updated matrix will all be 1, resulting in a full-rank matrix. However, we are still able to perform feature selection in this case by selecting the r largest eigenvalues, similar to feature selection in principal components analysis (PCA).

The complete algorithm is described below.

Algorithm 1. Mirror Descent for Metric Learning

input: data $(\mathbf{x}_t, \mathbf{z}_t, y_t)_{t=1}^T$, parameters ρ, $\eta > 0$
choose: Bregman functions $\psi(M)$; $\psi(\mu)$, loss function $\ell(M, \mu; \mathbf{x}, \mathbf{z}, y)$
initialize: $M_0 = I_n$, $\mu_0 = 1$
for $(\mathbf{x}^t, \mathbf{z}_t, y_t)$ **do**
  let $\mathbf{u}_t = \mathbf{x}_t - \mathbf{z}_t, \quad \eta_t = \eta/\sqrt{t}$
  compute gradients of loss $\nabla_M \ell_t = \alpha_t \mathbf{u}_t \mathbf{u}_t'$ and $\nabla_\mu \ell_t = -\alpha_t$ (see Table 1)
  write $\nabla\psi(M_t) = V_t \nabla\psi(\Lambda_t) V_t'$
  compute symmetric rank-one update $V_{t+1}\, \Lambda_{t+1}\, V_{t+1}' = V_t \nabla\psi(\Lambda_t) V_t' - \alpha \mathbf{u}_t \mathbf{u}_t'$
  shrink the eigenvalues $M_{t+1} = V_{t+1}\, \nabla\psi^{-1}\left(S_{\eta\rho}(\Lambda_{t+1})\right) V_{t+1}'$
  margin update $\mu_{t+1} = \max\left(\nabla\psi^{-1}\left(\nabla\psi(\mu_t) - \eta\, \nabla\ell_t(M_t, \mu_t)\right),\ 1\right)$
end for

5 Kernel MDML

There are two primary approaches to kernelizing metric learning algorithms: one based on the direct application of the kernel trick, and the other based on the application of the Kernel Principal Components Analysis (KPCA) framework [16]. We use the first approach here. Consider the (possibly infinite-dimensional) nonlinear mapping $\phi : \mathbb{X} \to \mathbb{F}$, that maps all data $\mathbf{x}$ in the input space $\mathbb{X}$ to a high-dimensional feature space $\mathbb{F}$. Associated with this map is a kernel function $\kappa(\cdot, \cdot)$ that can compute inner-products in $\mathbb{F}$ without explicit transformation. Let $X \in \mathbb{R}^{\ell \times n}$ denote the matrix of all examples and Φ denote the matrix

of corresponding high-dimensional vectors obtained from applying the mapping ϕ to the data. In *feature space*, the squared Mahalanobis distance is computed as $d^2(\phi(\mathbf{x}),\, \phi(\mathbf{z})) \;=\; \|L(\phi(\mathbf{x}) - \phi(\mathbf{z}))\,\|_2^2 \;=\; (\phi(\mathbf{x}) - \phi(\mathbf{z}))' L' L (\phi(\mathbf{x}) - \phi(\mathbf{z}))$. If we parameterize $L' \;=\; \Phi G'$, we have that

$$d^2(\phi(\mathbf{x}),\, \phi(\mathbf{z})) \;=\; (\phi(\mathbf{x}) - \phi(\mathbf{z}))' \Phi G' G \Phi (\phi(\mathbf{x}) - \phi(\mathbf{z})).$$

This allows us to kernelize the equation above which leads to a metric in the feature space, d_κ, expressed in terms of input-space vectors as:

$$d_\kappa^2(\mathbf{x},\, \mathbf{z}) \;=\; (\boldsymbol{\kappa}(X, \mathbf{x}) - \boldsymbol{\kappa}(X, \mathbf{z}))'\, M\, (\boldsymbol{\kappa}(X, \mathbf{x}) - \boldsymbol{\kappa}(X, \mathbf{z})), \tag{14}$$

where $\boldsymbol{\kappa}(X, \mathbf{x})$ is the column of the kernel matrix corresponding to $\mathbf{x}$, and where, with a slight abuse of notation, we have set $M = G'G$. Now, the margin in feature space can be redefined as,

$$m_\kappa(\mathbf{x}_t, \mathbf{z}_t, y_t) \;=\; y_t \,(\, \mu \;-\; (\boldsymbol{\kappa}(X, \mathbf{x}) - \boldsymbol{\kappa}(X, \mathbf{z}))'\, M\, (\boldsymbol{\kappa}(X, \mathbf{x}) - \boldsymbol{\kappa}(X, \mathbf{z}))\,)\,. \tag{15}$$

As before, we can define $\mathbf{u}_t = \boldsymbol{\kappa}(X, \mathbf{x}) - \boldsymbol{\kappa}(X, \mathbf{z})$. Finally, once the matrix M is learned, the Mahalanobis distance of some test point $\tilde{\mathbf{x}}$ with respect to a data point $\mathbf{x}$ can easily be computed as:

$$d_\kappa^2(\tilde{\mathbf{x}},\, \mathbf{x}) \;=\; \boldsymbol{\kappa}(X, \tilde{\mathbf{x}}) - \boldsymbol{\kappa}(X, \mathbf{x}))'\, M\, (\boldsymbol{\kappa}(X, \tilde{\mathbf{x}}) - \boldsymbol{\kappa}(X, \mathbf{x}). \tag{16}$$

6 Related Work

Prior approaches to learning the Mahalanobis metric include work by Xing et al., [5], the SVM-based approach of Schultz and Joachims [26] and large-margin nearest neighbors (LMNN) [6]. Davis et al., formulate the metric learning problem as minimizing the Burg divergence subject to similarity constraints, an approach called information theoretic metric learning (ITML) [7]. The BoostMetric approach developed by Shen et al., generalizes the well known AdaBoost algorithm to use a metric as a weak learner rather than a classifier [8]. This results in the optimization of the exponential loss of the margin function, which is solved via coordinate descent. Recently, Guillaumin et al., [27] proposed a metric learning approach that uses logistic regression loss. Many of these algorithms can be viewed as cases of the MDML approach presented here. The MDML approach is also closely related to low rank kernel learning, which was studied by Kulis et al., [25], where the nearness of kernels is measured using the von Neumann and Burg divergences.

There also exist several metric learning approaches such as discriminant adaptive nearest neighbor classification (DANN) [28], neighborhood components analysis (NCA) [29] and relevant components analysis (RCA) [30] that can perform feature selection, in addition to learning a metric. Other online algorithms for supervised learning of the Mahalanobis metric include the work of Shalev-Shwartz et al., [9] and Jain et al, [31].

Table 2. UCI data sets

Data set	#train	#test	#dim	#trn pairs	# classes
iris	105	45	4	630	3
wine	123	55	13	738	3
scale	436	189	4	2616	3
segment	147	63	19	882	7
breast	397	172	30	2382	2
ionosphere	245	106	34	1470	2

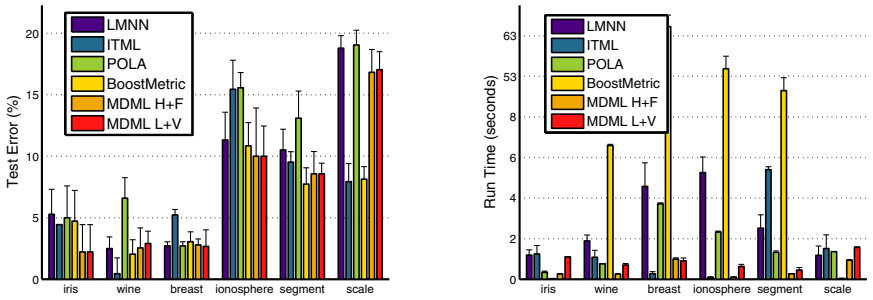


Fig. 3. Comparing test error (**left**) and run times (**right**) on six UCI data sets

7 Experiments

In this section, we compare the MDML approach with some current metric learning approaches on various data sets. We consider two classes of algorithms: an additive algorithm that arises from using the hinge loss with the Frobenius divergence (MDML H+F) and a multiplicative algorithm that arises from using the logistic loss with the von Neumann divergence (MDML L+V).

7.1 Benchmark Data Sets

We consider four well-known batch and online metric learning approaches: LMNN[3], ITML[4], BoostMetric[5] and POLA [9]. The latter, as well as the MDML approaches were implemented in `MATLAB`.

The performance of these methods on six data sets from the UCI repository[6]. The statistics of these data sets are described in Table 2. All data sets were normalized to zero mean and unit standard deviation. The experimental results are averaged over 10 runs; for each run, the data was split uniformly randomly into training and test sets: 70% was used for training and the remaining 30%

[3] http://www.cse.wustl.edu/~kilian/code/code.html
[4] http://www.cs.utexas.edu/~pjain/itml/
[5] http://code.google.com/p/boosting/
[6] http://archive.ics.uci.edu/ml/

was used for testing. The various parameters in ITML, LMNN, MDML H+F and MDML L+V were selected through 10-fold cross validation.

For each data set, we generate triplets and similar/dissimilar pairs for the methods here based on the approach described by Weinberger et al., [6]. To summarize, for each data point $\mathbf{x}_t$, k similarly labeled nearest neighbors (targets) and k differently labeled nearest neighbors (impostors) are selected, and triplets are constructed appropriately. We chose $k = 3$ for the generation of triplets and labeled pairs. Once the models were learned, test data were classified using 3-nearest neighbors classification as well.

Figure 3 shows the performance of these approaches with respect to test error and running time.The generalization performance of the MDML approaches is consistently comparable to that of other metric learning approaches. However, the significance of the MDML approaches becomes apparent when considering the running times. BoostMetric is the most expensive approach here, even among the batch approaches, which is not surprising considering it is an ensemble approach. The generalization performance of BoostMetric is good overall, but the performance comes at a significantly higher computational cost. While the MDML approaches outperform the batch methods computationally, they are also faster than POLA, which is an online approach.

We also studied the feature selection performance of the MDML approaches on these benchmark datasets. These results are shown in Figure 4. While it is clear that increasing values of ρ force more features to zero, it is interesting to note that, in many cases, with an appropriate choice of the learning rate η, it is possible to learn a highly sparse metric whose generalization performance does not degrade significantly. As hoped, the algorithm performs input-space feature selection while learning a pseudo-metric, which can be helpful to practitioners when trying to learn interpretable models. A similar trend is observed when counting the number of eigenvalues that account for 90% of the cumulative energy of the metric. Again, the MDML approaches are able to accumulate more information into a smaller subset of features. While Boostmetric and LMNN are able to perform well by this measure, it should be noted again, that this comes at a higher computational expense.

7.2 Digit Recognition

We use the Optical Recognition of Handwritten Digits (`optdigits`) data set from the UCI repository for these experiments. This 64-dimensional, 10 class data set consists of 3823 training points and 1797 test points. Generating triplets using the approach by Weinberger et al., as described above, results in $34,407$ triplets for training. For the metric learning methods that take labeled pairs, this approach resulted in the generation of $11,469$ similar pairs of data and $11,469$ dissimilar pairs of data. As before, we set $k = 3$ for both training and testing. Parameters were selected using 5-fold cross validation. The results are summarized in Table 3.

We compare the different approaches on test error, run time and feature selection. As with the previous benchmark results, LMNN and BoostMetric are

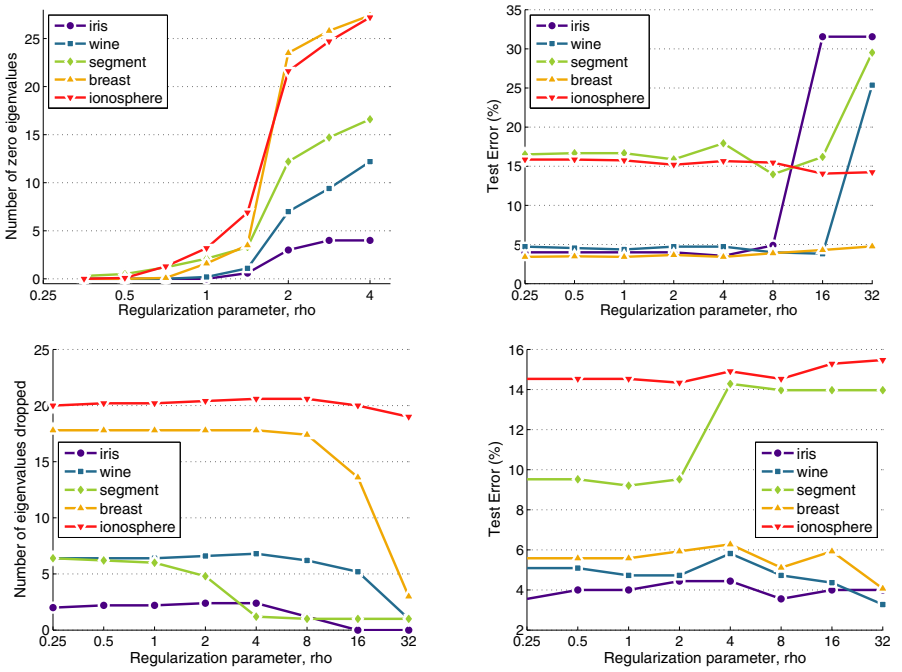


Fig. 4. (**top**) `MDML H+F`: (**left**) Average number of zero eigenvalues of the learned metric, with fixed learning rate η, and different regularization parameters ρ; (**right**) the corresponding test error. (**bottom**) `MDML L+V`: number of features dropped i.e., whose eigenvalues do not contribute to the top 90% of the cumulative energy. This experiment could not be performed for `scale` as it was extremely sensitive to parameters.

able to produce the best models, but again, this comes at the expense of a large computational cost, particularly in the case of BoostMetric. The MDML approaches are able to generalize well overall, but the overall run time for both methods is several orders of magnitude smaller. We also compare the ability to perform feature selection across all the data sets using two measures. Given $L = V\mathsf{diag}(\boldsymbol{\lambda})V'$, the first measure is simply the number of non-zero eigenvalues of L i.e., $\|\boldsymbol{\lambda}\|_0$. The second measure is the number of eigenvalues required to account for 90% of the cumulative energy of the metric. The cumulative energy of the i-th largest eigenvalue is $e_i = \sum_{j=1}^{i} \lambda_j$. The last column shows the number of features r such that $e_r \geq 0.9 \sum_{i=1}^{n} \lambda_i$; this is used to pick a reduced subset of features during PCA. Both MDML methods are able to perform input space feature selection effectively; for von Neumann, even though a full-rank matrix is learned, we are able to pick a reduced subset because the cumulative energy is concentrated in a few eigenvalues.

Table 3. Performance of the different approaches on the `optdigits` data set

Data set	Test Error (%)	Run Time (seconds)	Non-zero features	Num. feats. for 90% energy
`LMNN`	1.669	54.213	30	20
`ITML`	5.509	25.745	62	43
`POLA`	2.282	14.607	53	40
`BoostMetric`	1.758	2072.427	62	19
`MDML H+F`	1.892	15.232	26	22
`MDML L+V`	1.948	13.768	62	29

8 Conclusions and Future Work

We have presented an incremental metric learning approach (MDML) which not only optimizes the notion of loss at every step, but is also regularized. Specifically, we are interested in learning metrics that are sparse in the eigenspectrum, and to this end, the metric learning problem was regularized with the trace norm. This formulation is solved using composite mirror descent and results in a very general framework. Several different types of algorithms can be derived by choosing different loss functions and Bregman functions. Furthermore, the updates result in a symmetric rank-one update at the current step; this can be implemented very efficiently making the approach scalable to large data sets. Preliminary experimental results suggest that the approach performs comparably with current approaches with regard to generalization. However, the ability to learn a metric along with feature selection makes this approach attractive to machine learning practitioners.

These proof-of-concept results suggests several exciting directions for future research, some of which are currently under consideration. Given that the updates are embarrassingly parallelizable, an immediate target is the massive data setting, where we need to learn with millions of data points. In addition, the approach is also amenable to the addition of local geometry constraints in order to learn low-dimensional geometry-aware metrics that lead to representable models. Finally, the kernel-MDML approach is a very powerful extension to linear metric learning, with applications in colored dimensionality reduction and manifold alignment.

Proof of Proposition 1

The optimal solution $\bar{M} = \bar{V}\,\mathsf{diag}(\bar{\boldsymbol{\lambda}})\,\bar{V}'$ should satisfy the gradient condition for (10):

$$\nabla\psi(M_{t+\frac{1}{2}}) - \nabla\psi(\bar{M}) \;\in\; \eta\rho\,\partial\,|||\,\bar{M}|||, \tag{17}$$

where $\partial\,|||\,\bar{M}|||$ denotes the set of all subgradients of $|||\bar{M}|||$. For an $m \times n$ matrix, $\bar{M}$ with SVD of $\bar{M} = U\mathsf{diag}(\boldsymbol{\sigma})V'$ the subgradients are given by [32]:

$$\partial\;|||\,\bar{M}||| \;=\; \left\{UV' + W \,|\, W \in \mathbb{R}^{m\times n},\; U'W = 0,\; WV = 0,\; \|W\|_2 \le 1\right\}. \tag{18}$$

In this case, since $\bar{M}$ is symmetric, we have $\bar{M} = \bar{V}\bar{\Lambda}\bar{V}'$ and we now need to show that any $\bar{M}$ which satisfies $\bar{M} \succeq 0$ and the subgradient condition

$$\partial \, |||\, \bar{M}||| \;=\; \left\{ I_n + W \,|\, W \in \mathbb{S}^n,\; \bar{V}'W = 0,\; W\bar{V} = 0,\; \|W\|_2 \leq 1 \right\} \tag{19}$$

is optimal. Decompose $\nabla\psi(M_{t+\frac{1}{2}}) = V\,\Lambda\,V'$ further into $\nabla\psi(M_{t+\frac{1}{2}}) = V_+\Lambda_+V_+' + V_-\Lambda_-V_-'$, where the subscript $+$ (and similarly $-$) refers to components of V and Λ with eigenvalues $\lambda_i > \eta\rho$ (and similarly $\lambda_i \leq \eta\rho$). The gradient at the optimal solution $\nabla\psi(\bar{M}) = S_{\eta\rho}(\nabla\psi(M_{t+\frac{1}{2}}))$ can be expressed by directly thresholding the eigenvalues:

$$\nabla\psi(\bar{M}) \;=\; V_+(\Lambda_+ - \eta\rho\, I_n)V_+'.$$

Substituting the gradients into the first-order condition (17), we have

$$\begin{aligned} \nabla\psi(M_{t+\frac{1}{2}}) - \nabla\psi(\bar{M}) &= \eta\rho\, I_n \;+\; V_-\Lambda_-V_-', \\ &= \eta\rho\, \left(I_n \;+\; \tfrac{1}{\eta\rho}\, V_-\Lambda_-V_-' \right). \end{aligned} \tag{20}$$

Choosing $\bar{V} = V_+$ and $W = \frac{1}{\eta\rho}\, V_-\Lambda_-V_-'$, we immediately have $W \in \mathbb{S}^n$, $\bar{V}'W = 0$ and $W\bar{V} = 0$. Also, $\|W\|_2 = \eta\rho\, \|\boldsymbol{\lambda}_-\|_2 \leq 1$, since, by construction, all components of $\boldsymbol{\lambda}_-$ are $\leq \eta\rho$. By the very same construction, we also have that $\bar{M}$ is psd, since $\boldsymbol{\lambda}_+ > \eta\rho$. Thus, we have shown that $\nabla\psi(\bar{M}) = S_{\eta\rho}(\nabla\psi(M_{t+\frac{1}{2}}))$ is optimal to (10). $\square$

Acknowledgements. The authors gratefully acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181, and the National Institutes of Health under the National Library of Medicine grant no. NLM R01-LM008796. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government. The authors would also like to acknowledge anonymous reviewers for their invaluable comments.

References

1. MacQueen, J.: On convergence of k-means and partitions with minimum average variance. Annals of Mathematical Statistics 36, 1084 (1965)
2. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory IT-13(1) , 21–27 (1967)
3. Cox, T.F., Cox, M.A.A.: Multidimensional Scaling. Chapman and Hall (2001)
4. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: Proc. 18th ICML, pp. 577–584 (2001)
5. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: NIPS, vol. 15, pp. 505–512 (2002)
6. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: NIPS, vol. 19 (2006)
7. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: Proc. 24th ICML, pp. 209–216 (2007)
8. Shen, C., Kim, J., Wang, L., van den Hengel, A.: Positive semidefinite metric learning with boosting. In: NIPS, vol. 22, pp. 629–633 (2009)

9. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: Proc. 21st ICML, pp. 94–102 (2004)
10. Chopra, S., Hadsell, R., Lecun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: Proceedings of Computer Vision and Pattern Recognition Conference, pp. 539–546. IEEE Press (2005)
11. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B 58, 267–288 (1994)
12. Recht, B., Fazel, M., Parrilo, P.A.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Rev. 52(3), 471–501 (2010)
13. Bregman, L.M.: The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. USSR Computational Mathematics and Mathematical Physics 7, 200–217 (1967)
14. Beck, A., Teboulle, M.: Mirror descent and nonlinear projected subgradient methods for convex optimization. Operations Research Letters 31, 167–175 (2003)
15. Duchi, J., Shalev-Shwartz, S., Singer, Y., Tewari, A.: Composite objective mirror descent. In: COLT, pp. 14–26 (2010)
16. Chatpatanasiri, R., Korsrilabutr, T., Tangchanachaianan, P., Kijsirikul, B.: On kernelization of supervised Mahalanobis distance learners. Computing Research Repoisitory (CoRR) abs/0804.1441 (2008)
17. Censor, Y.A., Zenios, S.A.: Parallel Optimization: Theory, Algorithms and Applications. Oxford University Press (1997)
18. Kakade, S.M., Shalev-Shwartz, S., Tewari, A.: On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization (preprint)
19. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press (1990)
20. Cai, J.F., Candès, E.J., Shen, Z.: A singular value thresholding algorithm for matrix completion. SIAM Journal on Optimization 20, 1956–1982 (2010)
21. Lewis, A.S.: The convex analysis of unitarily invariant matrix functions. Journal of Convex Analysis 2, 173–183 (1995)
22. Lewis, A.S.: The mathematics of eigenvalue optimization. Mathematical Programming 97, 155–176 (2003)
23. Golub, G.H., Van Loan, C.F.: Matrix Computations (Johns Hopkins Studies in Mathematical Sciences). 3rd edn. The Johns Hopkins University Press (1996)
24. Gu, M., Eisenstat, S.C.: A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. SIAM Journal on Matrix Analysis and Applications 15(4), 1266–1276 (1994)
25. Kulis, B., Sustik, M.A., Dhillon, I.S.: Low-rank kernel learning with bregman matrix divergences. J. Mach. Learn. Res. 10, 341–376 (2009)
26. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: NIPS, vol. 16 (2004)
27. Guillaumin, M., Verbeek, J.J., Schmid, C.: Is that you? metric learning approaches for face identification. In: ICCV, pp. 498–505 (2009)
28. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest neighbor classification. IEEE Trans. on Pattern Analysis and Machine Intelligence 18(6), 607–616 (1996)
29. Goldberger, J., Roweis, S.T., Hinton, G.E., Salakhutdinov, R.: Neighbourhood components analysis. In: NIPS, vol. 17 (2004)
30. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning distance functions using equivalence relations. In: Proc. 20th ICML, pp. 11–18 (2003)
31. Jain, P., Kulis, B., Dhillon, I.S., Grauman, K.: Online metric learning and fast similarity search. In: NIPS, pp. 761–768 (2008)
32. Watson, G.A.: Characterization of the subdifferential of some matrix norms. Linear Algebra and its Applications 170, 33–45 (1992)

Author Index

Abbeel, Pieter I-1
Affenzeller, Michael II-824
Ahmadi, Babak I-585
Akaho, Shotaro II-35
Akrour, Riad II-116
Ali, Wafa Bel Haj I-314
An, Aijun II-483
Artières, T. I-506
Asoh, Hideki II-35
Atzmueller, Martin II-277, II-842
Aussem, Alex I-58

Bai, Bing I-159
Bai, Xiao I-207
Balcan, Maria-Florina II-846
Bannai, Hideo II-435
Barlaud, Michel I-314
Batal, Iyad II-260
Batina, Lejla I-253
Bauckhage, Christian II-850
Becker, Martin II-277
Beham, Andreas II-824
Bespalov, Dmitriy I-159
Bhargava, Aniruddha II-644
Bhattacharyya, Pushpak I-774
Bickel, Steffen II-676
Blanc, Sebastian II-828
Blasiak, Sam II-419
Blunsom, Phil I-760
Boden, Brigitte I-458
Böhm, Klemens II-828
Bootkrajang, Jakramate I-143
Boularias, Abdeslam II-227
Boullé, Marc II-243
Buccafurri, Francesco II-467
Buntine, Wray I-106
Burger, Thomas I-299
Burnside, Elizabeth S. I-617

Cao, Qiong I-283
Chang, Edward Y. I-553
Chang, Kuiyu I-794
Chen, Enhong II-548
Chen, Tianqi II-757
Cheng, Weiwei II-83
Chidlovskii, Boris I-681
Chin, Alvin II-613
Chowdhary, Girish II-99
Chu, Dejun I-537
Chudán, David II-808
Chung, Fu-lai II-789
Cissé, M. I-506
Cohen, William W. II-773
Collins, John II-132
Collins, Tony J. I-269
Cong, Gao I-569
Cooper, Gregory II-260
Courty, Nicolas I-299
Cox, James I-237
Crammer, Koby II-323

D'Ambrosio, Roberto I-314
Denoyer, Ludovic II-180
De Raedt, Luc I-2
Ding, Chris II-339
Diot, Fabien I-394
Dogan, Ürün I-122
Dubbin, Gregory I-760
Dugan, Michael II-51
Duh, Kevin II-293
Dulac-Arnold, Gabriel II-180
Duling, David I-237

Eck, Douglas I-4
Edelkamp, Stefan I-175
Elghazel, Haytham I-58
Elkan, Charles I-665, II-211
El-Yaniv, Ran I-744
Enokuma, Yuki II-435
Ermon, Stefano II-195, II-499

Faddoul, Jean Baptiste I-681
Faloutsos, Christos I-521
Faltings, Boi I-410
Forman, George II-51
Frasconi, Paolo II-854
Fries, Sergej I-458
Fromont, Elisa I-394

Gallinari, Patrick I-506, II-180
Gama, João I-827
Gao, Jing II-692
Garnett, Roman I-378
Gasse, Maxime I-58
Gay, Dominique II-243
Geramifard, Alborz II-99
Geurts, Pierre I-191, I-346
Ghahramani, Zoubin II-858
Ghosh, Shalini I-90
Giannotti, Fosca II-1
Gilleron, Rémi I-681
Glasmachers, Tobias I-122
Golshan, Behzad II-660
Gomes, Carla II-195
Görnitz, Nico I-633
Grabocka, Josif II-725
Graepel, Thore I-106
Guo, Shengbo I-106
Guo, Wenge II-1
Guo, Yuhong II-355
Gupta, Manish II-692

Hai, Phan Nhat II-820
Han, Jiawei II-692
Hancock, Edwin R. I-207
Harvey, Nicholas J.A. II-846
Hauskrecht, Milos II-260
Hazucha, Andrej II-808
Heyer, Gerhard II-812
Hidasi, Balázs II-67
Hinneburg, Alexander II-838
Ho, Tu Bao I-490
Hoi, Steven C.H. I-810
Hopcroft, John E. II-499
Hotho, Andreas I-728
How, Jonathan P. II-99, II-148
Hruschka Jr., Estevam R. II-307
Hu, Fangwei II-757
Huang, Kaizhu I-648
Huang, Sheng II-597
Huang, Tzu-Kuo II-741
Huang, Yi I-601
Hui, Siu Cheung I-794
Hüllermeier, Eyke II-83
Husaini, Mus'ab II-804

I, Tomohiro II-435
Ienco, Dino II-820
Igel, Christian I-122
Iwata, Tomoharu II-293

Jähnichen, Patrick II-812
Jeudy, Baptiste I-394
Jiang, Wenhao II-789
Jiang, Xueyan I-601

Kabán, Ata I-143
Kalousis, Alexandros I-223
Kamishima, Toshihiro II-35
Kandemir, Melih II-403
Kargar, Mehdi II-483
Kaski, Samuel II-403
Kautz, Henry I-90
Keim, Daniel I-5
Keller, Fabian II-828
Kersten, René I-42
Kersting, Kristian I-378, I-585, II-850
Ketter, Wolfgang II-132
Khardon, Roni I-711
Kibriya, Ashraf M. I-426
Kimura, Masahiro II-565
Kirshenbaum, Evan II-51
Klami, Arto II-403
Kliegr, Tomáš II-808
Kloft, Marius I-633
Kluegl, Peter I-728
Knobbe, Arno II-371
Koçyiğit, Ahmet II-804
Koedinger, Kenneth R. II-773
Kommenda, Michael II-824
Kong, Deguang II-339
Kong, Kang I-537
Kosina, Petr I-827
Kranjc, Janez II-816
Kriegel, Hans-Peter I-601
Krishnamurthy, Prashant II-531
Krömer, Oliver II-227
Kronberger, Gabriel II-824
Kumar, Abhishek I-665
Kunapuli, Gautam I-859
Kutzkov, Konstantin I-843

Laber, Eduardo S. II-709
Landwehr, Niels II-676
Laskey, Kathryn B. II-419
Lavrač, Nada II-816
Lax, Gianluca II-467
Lee, Sangkyun II-387

Lee, Wee-Sun II-164
Lemmerich, Florian I-728, II-277, II-842
Leong, Tze-Yun II-164
Li, Ming I-269
Li, Nan I-330, II-773
Li, Peng I-283
Li, Ping I-474
Li, Shaohua I-569
Liebig, Thomas II-629
Lijffijt, Jefrey II-451
Lippi, Marco II-854
Liu, Cheng-Lin I-648
Liu, Nathan N. II-757
Liu, Qi II-548
Liu, Ruilin II-1
Louppe, Gilles I-346
Luo, Zhigang I-269

Macropol, Kathy I-442
Maes, Francis I-191
Mantrach, Amin I-130
Marchiori, Elena I-253
Marek, Tomáš II-808
Marilly, Emmanuel I-394
Marteau, Pierre-François I-299
Martinot, Olivier I-394
Mavroeidis, Dimitrios I-253
May, Michael II-629
Menon, Aditya Krishna I-665
Miao, Chunyan I-569
Michini, Bernard II-148
Misra, Gaurav II-660
Monreale, Anna II-1
Morik, Katharina II-387
Motoda, Hiroshi II-565
Mücke, Manfred I-74
Mukherjee, Subhabrata I-774
Müller, Emmanuel II-828

Nanavati, Amit A. II-581
Nanopoulos, Alexandros II-725
Narayanam, Ramasuri II-581
Nassif, Houssam I-617
Natarajan, Sriraam I-585
Neumann, Marion I-378
Nguyen, Huy II-515
Nguyen, Tam T. I-794
Nguyen, Truong-Huy Dinh II-164
Nickel, Maximilian I-601
Niekler, Andreas II-812
Nielsen, Frank I-314
Nijssen, Siegfried II-371
Nocera, Antonino II-467
Nock, Richard I-314
Nowak, Robert II-644

Ohara, Kouzou II-565
Orbach, Matan II-323

Page, David I-617
Pápai, Tivadar I-90
Papalexakis, Evangelos E. I-521
Papapetrou, Panagiotis II-451
Passerini, Andrea II-854
Patricia, Novi I-378
Pedreschi, Dino II-1
Pelechrinis, Konstantinos II-531
Peng, Haoruo I-553
Pernkopf, Franz I-74
Peters, Jan II-227
Peters, Markus II-132
Podpečan, Vid II-816
Poncelet, Pascal II-820
Preiss, Rico II-838
Preux, Philippe II-180
Pu, Li I-410
Punta, Marco II-854
Puolamäki, Kai II-451
Puppe, Frank I-728

Qi, Yanjun I-159

Ramavajjala, Vivek II-211
Ramon, Jan I-362, I-426
Rangwala, Huzefa II-419
Rätsch, Gunnar I-633
Reinprecht, Peter I-74
Renders, Jean-Michel I-130
Reutemann, Peter II-833
Riondato, Matteo I-25

Saar-Tsechansky, Maytal II-132
Saito, Kazumi II-565
Sakuma, Jun II-35
Sanner, Scott I-106
Santos Costa, Vítor I-617
Sarle, Warren I-237
Sawade, Christoph II-676
Saygın, Yücel II-804
Scheffer, Tobias II-676

Scheibenpflug, Andreas II-824
Schmidt-Thieme, Lars II-725
Schneider, Jeff II-741
Schoenauer, Marc II-116
Schröder, René II-838
Schuurmans, Dale II-355
Sebag, Michèle II-116
Seidl, Thomas I-458
Selman, Bart II-195
Shad, Shafqat Ali II-548
Shavlik, Jude I-859
Shokoufandeh, Ali I-159
Shrivastava, Anshumali I-474
Sidiropoulos, Nicholas D. I-521
Siebes, Arno I-42
Šimůnek, Milan II-808
Singh, Ambuj I-442
Škrabal, Radek II-808
Smyth, Padhraic I-7
Sotelo, David II-709
Stolpe, Marco II-387
Stommel, Martin I-175
Sun, Yizhou II-692

Takeda, Masayuki II-435
Tang, Jie II-613
Tao, Qing I-537
Tapucu, Dilek II-804
Tatti, Nikolaj I-9
Teisseire, Maguelonne II-820
Terzi, Evimaria II-660
Than, Khoat I-490
Thurau, Christian II-850
Tikk, Domonkos II-67
Toepfer, Martin I-728
Tong, Hanghang II-597
Torre, Fabien I-681
Tresp, Volker I-601
Tschiatschek, Sebastian I-74

Upfal, Eli I-25
Ure, N. Kemal II-99
Ursino, Domenico II-467

Valentim, Caio II-709
van Laarhoven, Twan I-253
Vanschoren, Joaquin II-371, II-833
Vembu, Shankar I-665
Verma, Saurabh II-307
Vespier, Ugo II-371
Vetek, Akos II-403
Vinterbo, Staal A. II-19
Vojíř, Stanislav II-808
von Oertzen, Timo II-676
Vreeken, Jilles I-9

Wagner, Stefan II-824
Wahabzada, Mirwaes II-850
Wang, Hongnan I-269
Wang, Hui (Wendy) II-1
Wang, Jun I-223
Wang, Liaoruo II-499
Wang, Wei II-597, II-613
Wang, Xia II-613
Wang, Yuyang I-711
Wang, Yuyi I-362
Wang, Zhengyu I-553
Wehenkel, Louis I-191
Widmer, Christian I-633
Wilson, Andrew Gordon II-858
Woznica, Adam I-223
Wrobel, Stefan II-629
Wu, Gaowei I-537
Wu, Sen II-613

Xiang, Biao II-548
Xiao, Yanghua II-597
Xu, Bo II-597
Xu, Jun-Ming II-644
Xu, Tong II-548
Xu, Zhao II-629
Xue, Yexiang II-195

Yanay, David I-744
Yang, Deqing II-597
Yang, Peipei I-648
Yang, Qiang II-757
Yang, Yu II-548
Yanikoglu, Berrin II-804
Yeung, Dit-Yan I-697
Ying, Yiming I-283
Yu, Yang I-330
Yu, Yong II-757

Zhang, Xianglilan I-269
Zhang, Yu I-697
Zhang, Zhihong I-207
Zhang, Zhihua I-553
Zhao, Peilin I-810
Zhao, Zheng I-237

Zheng, Rong II-515
Zhou, Shuchang I-553
Zhou, Zhi-Hua I-330
Zhu, Xiaojin II-644
Zhuang, Honglei II-613
Zihayat, Morteza II-483