LNAI 6913

Dimitrios Gunopulos
Thomas Hofmann
Donato Malerba
Michalis Vazirgiannis (Eds.)

Machine Learning and Knowledge Discovery in Databases

European Conference, ECML PKDD 2011
Athens, Greece, September 2011
Proceedings, Part III

3 Part III

Springer

Lecture Notes in Artificial Intelligence 6913

Subseries of Lecture Notes in Computer Science

Dimitrios Gunopulos   Thomas Hofmann
Donato Malerba   Michalis Vazirgiannis (Eds.)

Machine Learning and Knowledge Discovery in Databases

European Conference, ECML PKDD 2011
Athens, Greece, September 5-9, 2011
Proceedings, Part III

Springer

Volume Editors

Dimitrios Gunopulos
University of Athens, Greece
E-mail: dg@di.uoa.gr

Thomas Hofmann
Google Switzerland GmbH, Zurich, Switzerland
E-mail: thofmann@google.com

Donato Malerba
University of Bari "Aldo Moro", Bari, Italy
E-mail: malerba@di.uniba.it

Michalis Vazirgiannis
Athens University of Economics and Business, Greece
E-mail: mvazirg@aueb.gr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-23807-9 e-ISBN 978-3-642-23808-6
DOI 10.1007/978-3-642-23808-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: Applied for

CR Subject Classification (1998): I.2, H.2.8, H.2, H.3, G.3, J.1, I.7, F.2.2, F.4.1

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)

Welcome to ECML PKDD 2011

Welcome to the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2011) held in Athens, Greece, during September 5–9, 2011. ECML PKDD is an annual conference that provides an international forum for the discussion of the latest high-quality research results in all areas related to machine learning and knowledge discovery in databases as well as other innovative application domains. Over the years it has evolved as one of the largest and most selective international conferences in machine learning and data mining, the only one that provides a common forum for these two closely related fields.

ECML PKDD 2011 included all the scientific events and activities of big conferences. The scientific program consisted of technical presentations of accepted papers, plenary talks by distinguished keynote speakers, workshops and tutorials, a discovery challenge track, as well as demo and industrial tracks. Moreover, two co-located workshops were organized on related research topics. We expect that all those scientific activities provide opportunities for knowledge dissemination, fruitful discussions and exchange of ideas among people both from academia and industry. Moreover, we hope that this conference will continue to offer a unique forum that stimulates and encourages close interaction among researchers working on machine learning and data mining.

We were very happy to have the conference back in Greece after 1995 when ECML was successfully organized in Heraklion, Crete. However, this was the first time that the joint ECML PKDD event was organized in Greece and, more specifically, in Athens, with the conference venue boasting a superb location under the Acropolis and in front of the Temple of Zeus. Besides the scientific activities, the conference offered delegates an attractive range of social activities, such as a welcome reception on the roof garden of the conference venue directly facing the Acropolis hill, a poster session at "Technopolis" Gazi industrial park, the conference banquet, and a farewell party at the new Acropolis Museum, one of the most impressive archaeological museums worldwide, which included a guided tour of the museum exhibits.

Several people worked hard together as a superb dedicated team to ensure the successful organization of this conference. First, we would like to express our thanks and deep gratitude to the PC Chairs Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba and Michalis Vazirgiannis. They efficiently carried out the enormous task of coordinating the rigorous hierarchical double-blind review process that resulted in a rich, while at the same time, very selective scientific program. Their contribution was crucial and essential in all phases and aspects of the conference organization and it was by no means restricted only to the paper review process. We would also like to thank the Area Chairs and Program Committee members for the valuable assistance they offered to the PC Chairs in their timely completion of the review process under strict deadlines. Special thanks should also be given to the Workshop Co-chairs, Bart Goethals and Katharina Morik, the Tutorial Co-chairs, Fosca Giannotti and Maguelonne Teisseire, the Discovery Challenge Co-chairs, Alexandros Kalousis and Vassilis Plachouras, the Industrial Session Co-chairs, Alexandros Ntoulas and Michail Vlachos, the Demo Track Co-chairs, Michelangelo Ceci and Spiros Papadimitriou, and the Best Paper Award Co-chairs, Sunita Sarawagi and Michèle Sebag. We further thank the keynote speakers, workshop organizers, the tutorial presenters and the organizers of the discovery challenge.

Furthermore, we are indebted to the Publicity Co-chairs, Annalisa Appice and Grigorios Tsoumakas, who developed and implemented an effective dissemination plan and supported the Program Chairs in the production of the proceedings, and also to Margarita Karkali for the development, support and timely update of the conference website. We further thank the members of the ECML PKDD Steering Committee for their valuable help and guidance.

The conference was financially supported by the following generous sponsors who are worthy of special acknowledgment: Google, Pascal2 Network, Xerox, Yahoo Labs, COST-MOVE Action, Rapid-I, FP7-MODAP Project, Athena RIC / Institute for the Management of Information Systems, Hellenic Artificial Intelligence Society, Marathon Data Systems, and Transinsight. Additional support was generously provided by Sony, Springer, and the UNESCO Privacy Chair Program. This support has given us the opportunity to specify low registration rates, provide video-recording services and support students through travel grants for attending the conference. The substantial effort of the Sponsorship Co-chairs, Ina Lauth and Ioannis Kopanakis, was crucial in order to attract these sponsorships, and therefore, they deserve our special thanks. Special thanks should also be given to the five organizing institutions, namely, University of Bari "Aldo Moro", Athens University of Economics and Business, University of Athens, University of Ioannina, and University of Piraeus for supporting in multiple ways our task.

We would like to especially acknowledge the members of the Local Organization team, Maria Halkidi, Despoina Kopanaki and Nikos Pelekis, for making all necessary local arrangements and Triaena Tours & Congress S.A. for efficiently handling finance and registrations. The essential contribution of the student volunteers also deserves special acknowledgment.

Finally, we are indebted to all researchers who considered this conference as a forum for presenting and disseminating their research work, as well as to all conference participants, hoping that this event will stimulate further expansion of research and industrial activities in machine learning and data mining.

July 2011

Aristidis Likas
Yannis Theodoridis

Preface

The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2011) took place in Athens, Greece, during September 5–9, 2011. This year we have completed the first decade since the junction between the European Conference on Machine Learning and the Principles and Practice of Knowledge Discovery in Data Bases conferences, which as independent conferences date back to 1986 and 1997, respectively. During this decade there has been an increasing integration of the two fields, as reflected by the rising number of submissions of top-quality research results. In 2008 a single ECML PKDD Steering Committee was established, which gathered senior members of both communities.

The ECML PKDD conference is a highly selective conference in both areas, the leading forum where researchers in machine learning and data mining can meet, present their work, exchange ideas, gain new knowledge and perspectives, and motivate the development of new interesting research results. Although traditionally based in Europe, ECML PKDD is also a truly international conference with rich and diverse participation.

In 2011, as in previous years, ECML PKDD followed a full week schedule, from Monday to Friday. It featured six plenary invited talks by Rakesh Agrawal, Albert-László Barabási, Christofer Bishop, Andrei Broder, Marco Gori and Heikki Mannila. Monday and Friday were devoted to workshops selected by Katharina Morik and Bart Goethals, and tutorials, organized and selected by Fosca Giannotti and Maguelonne Teisseire. There was also an interesting industrial session, managed by Alexandros Ntoulas and Michalis Vlachos, which welcomed distinguished speakers from the ML and DM industry: Vasilis Aggelis, Radu Jurca, Neel Sundaresan and Olivier Verscheure. The 2011 discovery challenge was organized by Alexandros Kalousis and Vassilis Plachouras.

The main conference program unfolded from Tuesday to Thursday, where 121 papers selected among 599 full-paper submissions were presented in the technical parallel sessions and in a poster session open to all accepted papers. The acceptance rate of 20% supports the traditionally high standards of the joint conference. The selection process was assisted by 35 Area Chairs, each supervising the reviews and discussions of about 17 papers, and by 270 members of the Program Committee, with the help of 197 additional reviewers. While the selection process was made particularly intense due to the very high number of submissions, we are grateful and heartily thank all Area Chairs, members of the Program Committee, and additional reviewers for their commitment and hard work during the tight reviewing period.

The composition of the paper topics covered a wide spectrum of issues. A significant portion of the accepted papers dealt with core issues such as supervised and unsupervised learning with some innovative contributions in fundamental issues such as cluster-ability of a dataset.

Other fundamental issues tackled by accepted papers include dimensionality reduction, distance and similarity learning, model learning and matrix/tensor analysis. In addition, there was a significant cluster of papers with valuable contributions on graph mining, graphical models, hidden Markov models, kernel methods, active and ensemble learning, semi-supervised and transductive learning, mining sparse representations, model learning, inductive logic programming, and statistical learning.

A significant part of the program covered novel and timely applications of data mining and machine learning in industrial domains, including: privacy-preserving and discrimination-aware mining, spatiotemporal data mining, text mining, topic modeling, learning from environmental and scientific data, Web mining and Web search, link analysis, bio/medical data, data Streams and sensor data, ontology-based data, relational data mining, learning from time series data, time series data.

In the past three editions of the joint conference, the two Springer journals *Machine Learning* and *Data Mining and Knowledge Discovery* published the top papers in two special issues printed in advance of the conference. These papers were not included in the conference proceedings, so there was no double publication of the same work. A novelty introduced this year was the post-conference publication of the special issues in order to guarantee the expected high-standard reviews for top-quality journals. Therefore, authors of selected machine learning and data mining papers were invited to submit a significantly extended version of their paper to the special issues. The selection was made by Program Chairs on the basis of their exceptional scientific quality and high impact on the field, as indicated by conference reviewers.

Following an earlier tradition, the Best Paper Chairs Sunita Sarawagi and Michèle Sebag contributed to the selection of papers deserving the Best Paper Awards and Best Student Paper Awards in Machine Learning and in Data Mining, sponsored by Springer. As ECML PKDD completes 10 years of joint organization, the PC chairs, together with the steering committee, initiated a 10-year Awards series. This award is established for the author(s), whose paper appeared in the ECML PKDD conference 10 years ago, and had the most impact on the machine learning and data mining research since then. This year's, first award, committee consisted of three PC co-chairs (Dimitrios Gunopulos, Donato Malerba and Michalis Vazirgiannis) and three Steering Committee members (Wray Buntine, Bart Goethals and Michèle Sebag).

The conference also featured a demo track, managed by Michelangelo Ceci and Spiros Papadimitriou; 11 demos out of 21 submitted were selected by a Demo Track Program Committee, presenting prototype systems that illustrate the high impact of machine learning and data mining application in technology. The demo descriptions are included in the proceedings. We further thank the members of the Demo Track Program Committee for their efforts in timely reviewing submitted demos.

Finally, we would like to thank the General Chairs, Aristidis Likas and Yannis Theodoridis, for their critical role in the success of the conference, the Tutorial, Workshop, Demo, Industrial Session, Discovery Challenge, Best Paper, and Local Chairs, the Area Chairs and all reviewers, for their voluntary, highly dedicated and exceptional work, and the ECML PKDD Steering Committee for their help and support. Our last and warmest thanks go to all the invited speakers, the speakers, all the attendees, and especially to the authors who chose to submit their work to the ECML PKDD conference and thus enabled us to build up this memorable scientific event.

July 2011

Dimitrios Gunopulos
Thomas Hofmann
Donato Malerba
Michalis Vazirgiannis

Organization

ECML/PKDD 2011 was organized by the Department of Computer Science—University of Ioannina, Department of Informatics—University of Piraeus, Department of Informatics and Telecommunications—University of Athens, Google Inc. (Zurich), Dipartimento di Informatica—Università degli Studi di Bari "Aldo Moro," and Department of Informatics—Athens University of Economics & Business.

General Chairs

Aristidis Likas	University of Ioannina, Greece
Yannis Theodoridis	University of Piraeus, Greece

Program Chairs

Dimitrios Gunopulos	University of Athens, Greece
Thomas Hofmann	Google Inc., Zurich, Switzerland
Donato Malerba	University of Bari "Aldo Moro," Italy
Michalis Vazirgiannis	Athens University of Economics and Business, Greece

Workshop Chairs

Katharina Morik	University of Dortmund, Germany
Bart Goethals	University of Antwerp, Belgium

Tutorial Chairs

Fosca Giannotti	Knowledge Discovery and Delivery Lab, Italy
Maguelonne Teisseire	TETIS Lab. Department of Information System and LIRMM Lab. Department of Computer Science, France

Best Paper Award Chairs

Sunita Sarawagi	Computer Science and Engineering, IIT Bombay, India
Michèle Sebag	University of Paris-Sud, France

Industrial Session Chairs

Alexandros Ntoulas	Microsoft Research, USA
Michail Vlachos	IBM Zurich Research Laboratory, Switzerland

Demo Track Chairs

Michelangelo Ceci	University of Bari “Aldo Moro,” Italy
Spiros Papadimitriou	Google Research

Discovery Challenge Chairs

Alexandros Kalousis	University of Geneva, Switzerland
Vassilis Plachouras	Athens University of Economics and Business, Greece

Publicity Chairs

Annalisa Appice	University of Bari “Aldo Moro,” Italy
Grigorios Tsoumakas	Aristotle University of Thessaloniki, Greece

Sponsorship Chairs

Ina Lauth	IAIS Fraunhofer, Germany
Ioannis Kopanakis	Technological Educational Institute of Crete, Greece

Organizing Committee

Maria Halkidi	University of Pireaus, Greece
Despina Kopanaki	University of Piraeus, Greece
Nikos Pelekis	University of Piraeus, Greece

Steering Committee

José Balcázar
Francesco Bonchi
Wray Buntine
Walter Daelemans
Aristides Gionis
Bart Goethals
Katharina Morik
Dunja Mladenic
John Shawe-Taylor
Michèle Sebag

Area Chairs

Elena Baralis
Hendrik Blockeel
Francesco Bonchi
Gautam Das
Janez Demsar
Amol Deshpande
Carlotta Domeniconi
Tapio Elomaa
Floriana Esposito
Fazel Famili
Wei Fan
Peter Flach
Johannes Furnkranz
Aristides Gionis
George Karypis
Ravi Kumar
James Kwok
Stan Matwin
Michael May
Taneli Mielikainen
Yücel Saygin
Arno Siebes
Jian Pei
Myra Spiliopoulou
Jie Tang
Evimaria Terzi
Bhavani M. Thuraisingham
Hannu Toivonen
Luis Torgo
Ioannis Tsamardinos
Panayiotis Tsaparas
Ioannis P. Vlahavas
Haixun Wang
Stefan Wrobel
Xindong Wu

Program Committee

Foto Afrati
Aijun An
Aris Anagnostopoulos
Gennady Andrienko
Ion Androutsopoulos
Annalisa Appice
Marta Arias
Ira Assent
Vassilis Athitsos
Martin Atzmueller
Jose Luis Balcazar
Daniel Barbara
Sugato Basu
Roberto Bayardo
Klaus Berberich
Bettina Berendt
Michele Berlingerio
Michael Berthold
Indrajit Bhattacharya
Marenglen Biba
Albert Bifet
Enrico Blanzieri
Konstantinos Blekas
Mario Boley
Zoran Bosnic
Marco Botta
Jean-Francois Boulicaut
Pavel Bradzil
Ulf Brefeld
Paula Brito
Wray Buntine
Toon Calders
Rui Camacho
Longbing Cao
Michelangelo Ceci
Tania Cerquitelli
Sharma Chakravarthy
Keith Chan
Vineet Chaoji
Keke Chen
Ling Chen
Xue-wen Chen
Weiwei Cheng
Yun Chi

Silvia Chiusano
Vassilis Christophides
Frans Coenen
James Cussens
Alfredo Cuzzocrea
Maria Damiani
Atish Das Sarma
Tijl De Bie
Jeroen De Knijf
Colin de la Higuera
Antonios Deligiannakis
Krzysztof Dembczynski
Anne Denton
Christian Desrosiers
Wei Ding
Ying Ding
Debora Donato
Kurt Driessens
Chris Drummond
Pierre Dupont
Saso Dzeroski
Tina Eliassi-Rad
Roberto Esposito
Nicola Fanizzi
Fabio Fassetti
Ad Feelders
Hakan Ferhatosmanoglou
Stefano Ferilli
Cesar Ferri
Daan Fierens
Eibe Frank
Enrique Frias-Martinez
Elisa Fromont
Efstratios Gallopoulos
Byron Gao
Jing Gao
Paolo Garza
Ricard Gavalda
Floris Geerts
Pierre Geurts
Aris Gkoulalas-Divanis
Bart Goethals
Vivekanand Gopalkrishnan
Marco Gori
Henrik Grosskreutz
Maxim Gurevich
Maria Halkidi
Mohammad Hasan
Jose Hernandez-Orallo
Eyke Hüllermeier
Vasant Honavar
Andreas Hotho
Xiaohua Hu
Ming Hua
Minlie Huang
Marcus Hutter
Nathalie Japkowicz
Szymon Jaroszewicz
Daxin Jiang
Alipio Jorge
Theodore Kalamboukis
Alexandros Kalousis
Panagiotis Karras
Samuel Kaski
Ioannis Katakis
John Keane
Kristian Kersting
Latifur Khan
Joost Kok
Christian Konig
Irena Koprinska
Walter Kosters
Georgia Koutrika
Stefan Kramer
Raghuram Krishnapuram
Marzena Kryszkiewicz
Nicolas Lachiche
Nada Lăvrac
Wang-Chien Lee
Feifei Li
Jiuyong Li
Juanzi Li
Tao Li
Chih-Jen Lin
Hsuan-Tien Lin
Jessica Lin
Shou-de Lin
Song Lin
Helger Lipmaa
Bing Liu

Huan Liu
Yan Liu
Corrado Loglisci
Chang-Tien Lu
Ping Luo
Panagis Magdalinos
Giuseppe Manco
Yannis Manolopoulos
Simone Marinai
Dimitrios Mavroeidis
Ernestina Menasalvas
Rosa Meo
Pauli Miettinen
Dunja Mladenic
Marie-Francine Moens
Katharina Morik
Mirco Nanni
Alexandros Nanopoulos
Benjamin Nguyen
Frank Nielsen
Siegfried Nijssen
Richard Nock
Kjetil Norvag
Irene Ntoutsi
Salvatore Orlando
Gerhard Paass
George Paliouras
Apostolos Papadopoulos
Panagiotis Papapetrou
Stelios Paparizos
Dimitris Pappas
Ioannis Partalas
Srinivasan Parthasarathy
Andrea Passerini
Vladimir Pavlovic
Dino Pedreschi
Nikos Pelekis
Jing Peng
Ruggero Pensa
Bernhard Pfahringer
Fabio Pinelli
Enric Plaza
George Potamias
Michalis Potamias
Doina Precup
Kunal Punera
Chedy Raissi
Jan Ramon
Huzefa Rangwala
Zbigniew Ras
Ann Ratanamahatana
Jan Rauch
Matthias Renz
Christophe Rigotti
Fabrizio Riguzzi
Celine Robardet
Marko Robnik-Sikonja
Pedro Rodrigues
Fabrice Rossi
Juho Rousu
Celine Rouveirol
Ulrich Rückert
Salvatore Ruggieri
Stefan Ruping
Lorenza Saitta
Ansaf Salleb-Aouissi
Claudio Sartori
Lars Schmidt-Thieme
Matthias Schubert
Michèle Sebag
Thomas Seidl
Prithviraj Sen
Andrzej Skowron
Carlos Soares
Yangqiu Song
Alessandro Sperduti
Jerzy Stefanowski
Jean-Marc Steyaert
Alberto Suarez
Johan Suykens
Einoshin Suzuki
Panagiotis Symeonidis
Marcin Szczuka
Andrea Tagarelli
Domenico Talia
Pang-Ning Tan
Letizia Tanca
Lei Tang
Dacheng Tao
Nikolaj Tatti

Martin Theobald
Dimitrios Thilikos
Jilei Tian
Ivor Tsang
Grigorios Tsoumakas
Theodoros Tzouramanis
Antti Ukkonen
Takeaki Uno
Athina Vakali
Giorgio Valentini
Maarten van Someren
Iraklis Varlamis
Julien Velcin
Celine Vens
Jean-Philippe Vert
Vassilios Verykios
Herna Viktor
Jilles Vreeken
Willem Waegeman
Jianyong Wang
Wei Wang
Xuanhui Wang
Hui Xiong

Jieping Ye
Jeffrey Yu
Philip Yu
Bianca Zadrozny
Gerson Zaverucha
Demetris Zeinalipour
Filip Zelezny
Changshui Zhang
Kai Zhang
Kun Zhang
Min-Ling Zhang
Nan Zhang
Shichao Zhang
Zhongfei Zhang
Junping Zhang
Ying Zhao
Bin Zhou
Zhi-Hua Zhou
Kenny Zhu
Xingquan Zhu
Djamel Zighed
Indre Zliobaite
Blaz Zupan

Additional Reviewers

Pedro Abreu
Raman Adaikkalavan
Artur Aiguzhinov
Darko Aleksovski
Tristan Allard
Alessia Amelio
Panayiotis Andreou
Fabrizio Angiulli
Josephine Antoniou
Andrea Argentini
Krisztian Balog
Teresa M.A. Basile
Christian Beecks
Antonio Bella
Aurelien Bellet
Dominik Benz
Thomas Bernecker
Alberto Bertoni
Jerzy Blaszczynski

Brigitte Boden
Samuel Branders
Janez Brank
Agnès Braud
Giulia Bruno
Luca Cagliero
Yuanzhe Cai
Ercan Canhas
Eugenio Cesario
George Chatzimilioudis
Xi Chen
Anna Ciampi
Marek Ciglan
Tyler Clemons
Joseph Cohen
Carmela Comito
Andreas Constantinides
Michael Corsello
Michele Coscia

Gianni Costa
Claudia D'Amato
Mayank Daswani
Gerben de Vries
Nicoletta Del Buono
Elnaz Delpisheh
Elnaz Delpisheh
Engin Demir
Nicola Di Mauro
Ivica Dimitrovski
Martin Dimkovski
Huyen Do
Lucas Drumond
Ana Luisa Duboc
Tobias Emrich
Saeed Faisal
Zheng Fang
Wei Feng
Cèsar Ferri
Alessandro Fiori
Nuno A. Fonseca
Marco Frasca
Christoph Freudenthaler
Sergej Fries
Natalja Friesen
David Fuhry
Dimitrios Galanis
Zeno Gantner
Bo Gao
Juan Carlos Gomez
Alberto Grand
Francesco Gullo
Tias Guns
Marwan Hassani
Zhouzhou He
Daniel Hernàndez-Lobato
Tomas Horvath
Dino Ienco
Elena Ikonomovska
Anca Ivanescu
Francois Jacquenet
Baptiste Jeudy
Dustin Jiang
Lili Jiang
Maria Jose
Faisal Kamiran
Michael Kamp
Mehdi Kargar
Mehdi Kaytoue
Jyrki Kivinen
Dragi Kocev
Domen Košir
Iordanis Koutsopoulos
Hardy Kremer
Anastasia Krithara
Artus Krohn-Grimberghe
Onur Kucuktunc
Tor Lattimore
Florian Lemmerich
Jun Li
Rong-Hua Li
Yingming Li
Siyi Liu
Stefano Lodi
Claudio Lucchese
Gjorgji Madjarov
M. M. Hassan Mahmud
Fernando Martínez-Plumed
Elio Masciari
Michael Mathioudakis
Ida Mele
Corrado Mencar
Glauber Menezes
Pasquale Minervini
Ieva Mitasiunaite-Besson
Folke Mitzlaff
Anna Monreale
Gianluca Moro
Alessandro Moschitti
Yang Mu
Ricardo Ñanculef
Franco Maria Nardini
Robert Neumayer
Phuong Nguyen
Stefan Novak
Tim O'Keefe
Riccardo Ortale
Aline Paes
George Pallis
Luca Pappalardo

Jérôme Paul
Daniel Paurat
Sergios Petridis
Darko Pevec
Jean-Philippe Peyrache
Anja Pilz
Marc Plantevit
Matija Polajnar
Giovanni Ponti
Adriana Prado
Xu Pu
ZhongAng Qi
Ariadna Quattoni
Alessandra Raffaetà
Maria Jose Ramírez-Quintana
Hongda Ren
Salvatore Rinzivillo
Ettore Ritacco
Matthew Robards
Christophe Rodrigues
Giulio Rossetti
André Rossi
Cláudio Sá
Venu Satuluri
Leander Schietgat
Marijn Schraagen
Wen Shao
Wei Shen
A. Pedro Duarte Silva
Claudio Silvestri
Ivica Slavkov
Henry Soldano
Yi Song
Miha Stajdohar
Anže Staric
Erik Strumbelj
Ilija Subasic
Peter Sunehag
Izabela Szczech
Frank Takes
Aditya Telang
Eleftherios Tiakas
Gabriele Tolomei
Marko Toplak
Daniel Trabold
Roberto Trasarti
Paolo Trunfio
George Tsatsaronis
Guy Van den Broeck
Antoine Veillard
Mathias Verbeke
Jan Verwaeren
Alessia Visconti
Dimitrios Vogiatzis
Dawei Wang
Jun Wang
Louis Wehenkel
Adam Woznica
Ming Yang
Qingyan Yang
Xintian Yang
Lan Zagar
Jure Žbontar
Bernard Zenko
Pin Zhao
Yuanyuan Zhu
Albrecht Zimmermann
Andreas Züfle

Sponsoring Institutions

We wish to express our gratitude to the sponsors of ECML PKDD 2011 for their essential contribution to the conference: Google, Pascal2 Network, Xerox, Yahoo Lab, COST/MOVE (Knowledge Discovery from Movin, Objects), FP7/MODAP (Mobility, Data Mining, and Privacy), Rapid-I (report the future), Athena/IMIS (Institute for the Management of Information Systems), EETN (Hellenic Artificial Intelligence Society), MDS Marathon Data Systems, Transinsight, SONY, UNESCO Privacy Chair, Springer, Università degli Studi di Bari "Aldo Moro," Athens University of Economics and Business, University of Ioannina, National and Kapodistrian University of Athens, and the University of Piraeus.

Table of Contents – Part III

Regular Papers

Demo Papers

Sparse Kernel-SARSA(λ) with an Eligibility Trace

Matthew Robards[1,2], Peter Sunehag[2], Scott Sanner[1,2], and Bhaskara Marthi[3]

[1] National ICT Australia
Locked Bag 8001
Canberra ACT 2601, Australia
[2] Research School of Computer Science
Australian National University
Canberra, ACT, 0200, Australia
[3] Willow Garage, Inc.,
68 Willow Road, Menlo Park, CA 94025, USA

Abstract. We introduce the *first* online kernelized version of SARSA(λ) to permit sparsification for arbitrary λ for $0 \leq \lambda \leq 1$; this is possible via a novel kernelization of the eligibility trace that is maintained separately from the kernelized value function. This separation is crucial for preserving the functional structure of the eligibility trace when using sparse kernel projection techniques that are essential for memory efficiency and capacity control. The result is a simple and practical Kernel-SARSA(λ) algorithm for general $0 \leq \lambda \leq 1$ that is memory-efficient in comparison to standard SARSA(λ) (using various basis functions) on a range of domains including a real robotics task running on a Willow Garage PR2 robot.

1 Introduction

In many practical reinforcement learning (RL) problems, the state space $\mathcal{S}$ may be very large or even continuous, leaving function approximation as the only viable solution. Arguably, the most popular form of RL function approximation uses a linear representation $\langle \mathbf{w}, \phi(s) \rangle$; although linear representations may seem quite limited, extensions based on kernel methods [14,2,13] provide (explicitly or implicitly) a rich feature map ϕ that in some cases permits the approximation of arbitrarily complex, nonlinear functions.

The simplicity and power of kernel methods for function approximation has given rise to a number of kernelized RL algorithms in recent years, e.g., [20,4,7,8,17]. In this paper we focus on online kernel extensions of SARSA(λ) — a *model-free* algorithm for learning optimal control policies in RL. However, unlike the Gaussian Process SARSA (GP-SARSA) approach and other kernelized extensions of SARSA [20,4], the main contribution of this paper is the *first* kernelized SARSA algorithm to allow for general $0 \leq \lambda \leq 1$ rather than restricting to just $\lambda \in \{0, 1\}$.

While generalizing an online kernelized SARSA algorithm to SARSA(λ) with $0 \leq \lambda \leq 1$ might seem as simple as using an eligibility trace [15], this leads

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 1–17, 2011.

to theoretical and practical issues when sparsity is introduced. Because online kernel methods typically require caching *all* previous data samples, sparsification techniques that selectively discard cached samples and reproject the value representation are necessary [3,5,9]. However, in kernelized SARSA(λ), cached samples are required for both the value *and* eligibility trace representation, hence sparsification that focuses on low-error value approximations may inadvertently destroy structure in the eligibility trace. To address these issues, we *separate* the kernelization of the value function from the eligibility function, thus maintaining independent low-error, sparse representations of each. Despite the mathematical complications, we derive simple and efficient value and eligbility updates under this scheme. We point out here that once one wishes to "switch off" the learning and utilize the learned policy, our algorithm becomes linear in the number of samples stored and our experimental section shows that our algorithm seems to be more efficient than regular SARSA (λ).

These novel insights allow us to propose a practical, memory-efficient Kernel-SARSA(λ) algorithm for general $0 \leq \lambda \leq 1$ that scales efficiently in comparison to standard SARSA(λ) (using both radial basis functions and tile coding) on a range of domains including a real robotics task running on a Willow Garage PR2 robot. Crucially we note the use of $0 < \lambda < 1$ often leads to the best performance in the fewest samples, indicating the importance of the two main paper contributions:

1. the first generalization of kernelized SARSA(λ) algorithms to permit $0 \leq \lambda \leq 1$ with sparsification, and
2. the novel kernelization and projection of *separate* value and eligibility functions needed for low-error, memory-efficient approximations of kernelized SARSA(λ) with an eligibility trace.

2 Preliminaries

In this section we briefly review [15] MDPs, the SARSA(λ) algorithm, its extension for function approximation and some background on Reproducing Kernel Hilbert Spaces [1].

2.1 Markov Decision Processes

We assume a (finite, countably infinite, or even continuous) Markov decision process (MDP) [11] given by the tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$. Here, states are given by $s \in \mathcal{S}$, actions are given by $a \in \mathcal{A}$, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function with $T(s, a, s')$ defining the probability of transitioning from state s to s' after executing action a. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function where $r_t = R(s_t, a_t, s_{t+1})$ is the reward received for time t after observing the transition from state s_t to s_{t+1} on action a_t. Finally, $0 \leq \gamma < 1$ is a discount factor.

A policy $\pi : \mathcal{S} \to \mathcal{A}$ specifies the action $\pi(s)$ to take in each state s. The value $Q_\pi(s, a)$ of taking an action a in state s and then following some policy π thereafter is defined using the infinite horizon, expected discounted reward criterion: $Q_\pi(s,a) = E_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot r_t | s_0 = s, a_0 = a]$. Our objective is to learn the optimal Q^* s.t. $\forall s, a, \pi$ $Q^*(s,a) \geq Q_\pi(s,a)$ in an episodic, online learning setting. Given Q^*, the optimal control policy π^* is simply $\pi^*(s) = \text{argmax}_a Q^*(s,a)$.

SARSA(λ) is a *temporal difference* RL algorithm for learning Q^* from experience [15]. We use SARSA(λ) in an *on-policy* manner where $Q_t(s,a)$ represents Q-value estimates at time t w.r.t. the greedy policy $\pi_t(s) := \text{argmax}_a Q_t(s,a)$. Initializing eligibilities $e_0(s,a) = 0$; $\forall s, a$, SARSA(λ) performs the following online Q-update at time $t+1$:

$$Q_{t+1}(s,a) = Q_t(s,a) + \eta err_t e_t(s,a); \, \forall s, a. \tag{1}$$

Here $\eta > 0$ is the learning rate, $err_t = Q_t(s_t,a_t) - R_t$ is the temporal difference error between the actual prediction $Q_t(s_t,a_t)$ and a bootstrapped estimate of $Q_t(s,a)$:

$$R_t = r_t + \gamma Q_t(s_{t+1}, a_{t+1})$$

and e_t is the *eligibility trace* updated each time step as follows:

$$e_{t+1}(s,a) = \begin{cases} \gamma\lambda e_t(s,a) + 1 & \text{if } s = s_t \text{ and } a = a_t \\ \gamma\lambda e_t(s,a) & \text{otherwise.} \end{cases} \tag{2}$$

The eligibility trace indicates the degree to which each state-action pair is updated based on future rewards. The parameter λ ($0 \leq \lambda \leq 1$) adjusts how far SARSA(λ) "looks" into the future when updating Q-values; as $\lambda \to 0$, SARSA(λ) updates become more myopic and it may take longer for delayed rewards to propagate back to earlier states.

For large or infinite state-action spaces it is necessary to combine SARSA(λ) with function approximation. Linear value approximation is perhaps the most popular approach: we let $\hat{Q}_t(s,a) = \langle \mathbf{w}_t, \phi(s,a) \rangle$ where $\mathbf{w}_t \in \mathbb{R}^d$ are $d > 0$ learned weights and $\phi : (s,a) \mapsto \phi(s,a)$ maps state-action (s,a) to features $\phi(s,a) \in \varPhi \subseteq \mathbb{R}^d$. Because the optimal Q_t may not exist within the span of $\hat{Q}_t$, we minimize the error between Q_t and $\hat{Q}_t$ in an online empirical risk minimization framework; this can be done by gradient descent on the squared error loss function $l[Q_t, s_t, R_t] = \frac{1}{2}(Q_t(s_t,a_t) - R_t)^2$ w.r.t. each observed datum (s_t, a_t, R_t). For SARSA(λ) with general λ and linear function approximation, Sutton and Barto [15] provide the following update rule

$$\mathbf{w}_{t+1} := \mathbf{w}_t + \eta err_t \mathbf{e}_t \phi(s_t, a_t) \tag{3}$$

with an eligibility vector updated through $\mathbf{e}_{t+1} = \gamma\lambda\mathbf{e}_t + \phi(s_t,a_t)$.

2.2 Reproducing Kernel Hilbert Spaces (RKHS)

When using Reproducing Kernel Hilbert Spaces [1] for function approximation (regression, classification) we define a feature map implicitly by defining a similarity measure called a kernel [14], [2], [13], e.g. a Gaussian kernel defined by $k(x,y) = e^{\|x-y\|^2/2\rho}$. Positive definite and symmetric kernels define a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ by completing the span of the functions

$k_x(\cdot) = k(x, \cdot)$ w.r.t. the inner product $\langle k_x, k_y \rangle_{\mathcal{H}} = k(x, y)$. Some kernels such as the Gaussian kernel are universal kernels, which means that the RKHS is dense in the space L^2 of square integrable functions. Using universal kernels means that any such (L^2) function can be approximated arbitrarily well by elements in $\mathcal{H}_k$.

If we have a feature map into a space with an inner product we have defined a kernel through $k(x, y) = \langle \phi(x), \phi(y) \rangle$. However, our intent is to start with a kernel like the Gaussian kernel and then use the feature map that is implicitly defined through that choice. This means that ϕ maps x to the function $k_x \in \mathcal{H}_k$. Note that $\phi(x)$ is not necessarily a finite vector anymore, but a possibly continuous function $k(x, \cdot)$.

3 Kernel-SARSA(λ)

We now generalize SARSA(λ) with function approximation to learn with large or even infinite feature vectors $\phi(s, a)$. We will use a reproducing kernel Hilbert space as our hypothesis space. The "weights" $\mathbf{w}$ that we will end up with are represented in the form $\mathbf{w} = \sum_i \alpha_i k((s_i, a_i), \cdot)$ and are really functions on $S \times A$. The corresponding Q function is

$$Q(s, a) = \sum_i \alpha_i k((s_i, a_i), (s, a)).$$

To define Kernel-SARSA(λ), we extend the SARSA (λ) update rule given in [15] to a RKHS setting. We slightly extend this update rule to include a regularizer term[1]. The update is given by

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \Big[(Q_t(s_t, a_t) - R_t)\mathbf{e}_t - \xi \mathbf{w}_t \Big] \tag{4}$$

where $\mathbf{e}_t$ is the eligibility trace, updated through

$$\mathbf{e}_t = \gamma \lambda \mathbf{e}_{t-1} + \phi(s_t, a_t), \tag{5}$$

s.t. $\phi(s_t, a_t) = k((s_t, a_t), \cdot)$ and $\mathbf{e}_t$ is initialized to $\mathbf{0}$ at the start of each episode. ξ denotes the regularizer. Alternatively we may write the eligibility trace as

$$\mathbf{e}_t = \sum_{i=t_0}^{t} (\gamma \lambda)^{t-i} \phi(s_i, a_i) \tag{6}$$

where t_0 is the time at which the current episode began. Typically such a representation would be undesirable since it requires storing all past samples, however

[1] Regularization is important for exact kernel methods that cache all samples since they have no other means of capacity control. Later when we introduce sparsification into the algorithm, the regularization term ξ may be set to zero since sparsity performs the role of capacity control.

kernelizing our online algorithm already necessitates storing all previously visited state-action pairs. Now, by substituting (6) into (4), we get

$$\mathbf{w}_{t+1} := \mathbf{w}_t - \eta \left(err_t \sum_{i=t_0}^{t} (\gamma\lambda)^{t-i} \phi(s_i, a_i) - \xi \mathbf{w}_t \right) \tag{7}$$

and assuming that $\mathbf{w}_0 = 0$ we see that

$$\mathbf{w}_t = \sum_{i=1}^{t-1} \alpha_i k((s_i, a_i), \cdot)$$

which leads us to an alternative formulation of (7). If err_t is the temporal difference error given by $(Q(s_t, a_t) - R_t))$ then

$$\sum_{i=1}^{t} \alpha_i k((s_i, a_i), \cdot) := \sum_{i=1}^{t-1} (1 - \eta\xi) \alpha_i k((s_i, a_i), \cdot) - \eta err_t \sum_{i=t_0}^{t} (\gamma\lambda)^{t-i} k((s_i, a_i), \cdot). \tag{8}$$

Equating the coefficients of the basis functions leads to the update formulae:

$$\alpha_i := (1 - \eta\xi)\alpha_i, \ i = 1, \ldots, t_0 - 1 \tag{9}$$

$$\alpha_t := \eta err_t \tag{10}$$

and otherwise for $i = t_0, ..., t - 1$

$$\alpha_i := (1 - \eta\xi)\alpha_i - \eta err_t (\gamma\lambda)^{t-i-1} \tag{11}$$

As a notational observation crucial for the next section, we note that $\mathbf{w}_t$ is exactly the same object as Q_t. Normally in function approximation one may think of $Q_t(\cdot)$ as $\langle \mathbf{w}_t, \phi(\cdot) \rangle$ which is true here, but since $\phi(s, a)(\cdot) = k((s, a), \cdot)$, we can conclude from the reproducing property that $\mathbf{w}_t(s, a) = \langle \mathbf{w}_t, \phi(s, a) \rangle$. We will henceforth write Q_t instead of $\mathbf{w}_t$ in our equations.

Although surprisingly simple and elegant, we note that this is the *first* kernelization of SARSA(λ) for general $0 \leq \lambda \leq 1$ that the authors are aware of. However, it is impractical in general since it requires storing all state-action pairs; next we provide a memory-efficient variant of this novel kernelized SARSA(λ) approach needed to make it practically viable.

4 Memory-Efficient Kernel-SARSA (λ) Based on the Projectron

We now have the foundations for a powerful kernel reinforcement learning algorithm. Problematically, however, the memory required to store the old samples grows without bound. We will deal with this by extending sparse representation techniques from [3,5,9].

Following the Projectron approach [9], in order to bound the memory requirements of the algorithm, we ask ourselves at each time step, "to what extent can the new sample be expressed as a linear combination of old samples?". Consider the "temporal hypothesis" Q'_t given through equation (7), and its projection $Q''_t = P_{t-1}Q'_t$ onto $\mathcal{H}_{t-1}$ which is the span of the set $\mathbb{S}$ of previously stored basis functions. One must be careful when trying to use (7) since our previous update equations made the vital assumption that we store all points allowing the progression from (5) to (6). This assumption no longer holds since we plan to only add those new points which cannot be well represented as a linear combination of the old ones. This is an obstacle that has to be resolved to be able to use the Projectron technique in our setting.

4.1 *Separately* Kernelizing the Eligibility Trace

We note that (6) represents the eligibility trace as a linear combination of previous basis functions. Hence we can write the eligibility trace (which is now really an eligibility function) as a function (separate from the value function) parameterized by $\beta = \{\beta_i\}_{i=1,\ldots,t}$ through

$$e_t = \sum_{i=1}^{t} \beta_i k((s_i, a_i), \cdot). \tag{12}$$

By substituting this form of the eligibility trace into its update equation (5) we get

$$\sum_{i=1}^{t} \beta_i k((s_i, a_i), \cdot) := \sum_{i=1}^{t-1} \gamma\lambda\beta_i k((s_i, a_i), \cdot) + k((s_t, a_t), \cdot) \tag{13}$$

and by equating the coefficients of the basis functions we get the parameter updates $\beta_i = \gamma\lambda\beta_i$ for $i = 1, \ldots, t-1$ and $\beta = 1$.

4.2 Projected Kernel-SARSA(λ) Updates

We begin by plugging the update of the eligibility trace into the Q update, and call this the temporal hypothesis given by

$$Q'_t = (1 - \eta\xi)Q_{t-1} - \eta err_t \Big[\gamma\lambda\mathbf{e}_{t-1} + k((s_t, a_t), \cdot)\Big] \tag{14}$$

allowing us to write its projection $Q''_t = P_{t-1}Q'_t =$

$$(1 - \eta\xi)Q_{t-1} - \eta err_t \Big[\gamma\lambda\mathbf{e}_{t-1} + P_{t-1}k((s_t, a_t), \cdot)\Big]. \tag{15}$$

Our aim is to examine how well the temporal hypothesis Q'_t is approximated by its projection onto $\mathcal{H}_{t-1}$ which suitably is the hypothesis in $\mathcal{H}_{t-1}$ closest to h. We denote the difference $Q''_t - Q'_t$ by δ_t and note that

$$\delta_t = -\eta err_t \Big[P_{t-1}k((s_t, a_t), \cdot) - k((s_t, a_t), \cdot)\Big]. \tag{16}$$

By letting $\mathbf{K}_{t-1}$ denote the kernel matrix with elements given by $\{\mathbf{K}_{t-1}\}_{i,j} = k((s_i, a_i), (s_j, a_j)$, $\mathbf{k}_t$ denote the vector with ith element $\mathbf{k}_{t_i} = k((s_i, a_i), (s_t, a_t))$ and letting $\mathbf{d}^* = \mathbf{K}_{t-1}^{-1}\mathbf{k}_t$ we can as in [9] derive that

$$\|\delta_t\|^2 = \eta^2 err_t^2[k((s_t, a_t), (s_t, a_t)) - \mathbf{k}_t^T\mathbf{d}^*]. \tag{17}$$

Now if $\|\delta_t\|^2$ is below some threshold ϵ, we update the Q function by setting it to

$$(1 - \eta\xi)Q_{t-1} - \eta err_t\left[\gamma\lambda\mathbf{e}_{t-1} + \sum_{i=1}^{|\mathbb{S}|}\mathbf{d}_i^* k((s_i, a_i), \cdot)\right]. \tag{18}$$

We note that the last part of Eq(18) is the projection of the eligibility trace given by

$$P_{t-1}e_t = \gamma\lambda e_{t-1} + \sum_{i=1}^{|\mathbb{S}|}\mathbf{d}_i^* k((s_i, a_i), \cdot) \tag{19}$$

$$= \gamma\lambda\sum_{i=1}^{|\mathbb{S}|}\beta_i k((s_i, a_i), \cdot) + \sum_{i=1}^{|\mathbb{S}|}\mathbf{d}_i^* k((s_i, a_i), \cdot) \tag{20}$$

giving the updates

$$\beta_i := \gamma\lambda\beta_i + \mathbf{d}_i^*,\ i = 1, ..., |\mathbb{S}|. \tag{21}$$

Finally we write Q_t and e_t in their parameterized form to obtain

$$\begin{aligned}\sum_{i=1}^{|\mathbb{S}|}\alpha_i k((s_i, a_i), \cdot) =& (1 - \eta\xi)\sum_{i=1}^{|\mathbb{S}|}\alpha_i k((s_i, a_i), \cdot) - \eta err_t\gamma\lambda P_{t-1}e_t \\ =& (1 - \eta\xi)\sum_{i=1}^{|\mathbb{S}|}\alpha_i k((s_i, a_i), \cdot) - \eta err_t\gamma\lambda\sum_{i=1}^{|\mathbb{S}|}\beta_i k((s_i, a_i), \cdot)\end{aligned} \tag{22}$$

and by again equating the coefficients of the basis functions we get the α update

$$\alpha_i = (1 - \eta\xi)\alpha_i - \eta err_t\gamma\lambda\beta_i \tag{23}$$

for $i = 1, \ldots, |\mathbb{S}|$ when $\delta_t < \epsilon$. Otherwise α is updated as in Equations (9)-(10). To avoid the costly calculation of the inverse kernel matrix we calculate this incrementally as in [9] when a new sample is added:

$$\mathbf{K}_t^{-1} = \begin{pmatrix} & & & 0 \\ & \mathbf{K}_{t-1}^{-1} & & \vdots \\ & & & 0 \\ 0 & \ldots & 0 & 0 \end{pmatrix} + \tag{24}$$

$$\frac{1}{k((s_t, a_t), (s_t, a_t)) - \mathbf{k}_t^T\mathbf{d}^*}\begin{pmatrix}\mathbf{d}^* \\ -1\end{pmatrix}\begin{pmatrix}\mathbf{d}^{*T} & -1\end{pmatrix}.$$

Algorithm 1. Memory Efficient Kernel-SARSA (λ)

```
INPUTS:
  – $\pi_0, \epsilon, \eta, \lambda$
  – $\mathbb{S} = \emptyset$

  1. DO
     (a) Select action $a_t$ (e.g. greedily) in current state $s_t$ and observe reward $r_t$
     (b) $\mathbf{d}^* \leftarrow \mathbf{K}_{t-1}^{-1}\mathbf{k}_t$
     (c) $\|\delta_t\|^2 \leftarrow \eta^2 err_t^2[k((s_t, a_t), (s_t, a_t)) - \mathbf{k}_t^T\mathbf{d}^*]$
     (d) if $(\|\delta_t\|^2 < \epsilon)$
           – for $i = 1, \dots, |\mathbb{S}|$
               • $\beta_i \leftarrow \gamma\lambda\beta_i + \mathbf{d}_i^*$
               • $\alpha_i \leftarrow (1 - \eta\xi)\alpha_i - \eta err_t \gamma\lambda\beta_i$
     (e) else
           – Add $k((s_t, a_t), \cdot)$ to $\mathbb{S}$
           – $\beta_{|\mathbb{S}|} \leftarrow 1$
           – for $i = 1, \dots, |\mathbb{S}| - 1$
               • $\beta_i \leftarrow \gamma\lambda\beta_i$
           – for $j = 1, \dots, |\mathbb{S}|$
               • $\alpha_i \leftarrow (1 - \eta\xi)\alpha_i - \eta err_t \gamma\lambda\beta_i$
           – Update $\mathbf{K}_{t-1}^{-1}$ through (24).
  2. UNTIL policy update required
```

From here on, references to Kernel-SARSA(λ) imply the memory-efficient version in Algorithm 1 and not the version in Section 3. This first version from Section 3 (a) cannot be used directly since it stores every sample, and (b) if sparsified naïvely via the Projectron, leads to an algorithm that stops learning long before convergence because most new kernel samples are discarded (since most lie within the span of previous samples) — unfortunately, the eligibility trace is defined in terms of these new samples! In this way, the eligibility trace is not updated and a directly sparsified approach to Section 3 will prematurely cease to learn, rendering it useless in practice.[2]

5 Empirical Evaluation

Having now completed the derivation of our memory efficient Kernel-SARSA(λ) algorithm, we proceed to empirically compare it to two of the most popular and useful function approximation approaches for SARSA(λ): one version using kernel (RBF) basis functions (n.b., not the same as Kernel-SARSA(λ) but with

[2] As such, this paper contributes much more than a simple combination of Section 3 and the Projectron [9] (which does not work) — it makes the crucial point that value function and eligibility function must be *separately* kernelized and projected as presented in Section 4.

similar function approximation characteristics) and the other using standard tile coding [15].

Our experimental objectives are threefold: (1) to show that Kernel-SARSA(λ) learns better with less memory than the other algorithms, (2) to show that $0 < \lambda < 1$ leads to optimal performance for Kernel-SARSA(λ) on each MDP, and (3) that Kernel-SARSA(λ) can learn a smooth nonlinear Q-function in a continuous space with less memory than competing algorithms *and* which is nearly optimal in a real-world domain.

5.1 Problems

We ran our algorithm on three MDPs: two standard benchmarks and one real-world robotics domain.

Pole balancing (cart pole): requires the agent to balance a pole hinged atop a cart by sliding the cart along a frictionless track. We refer to [15] for a specification of the transition dynamics; rewards are zero except for -1, which is received upon failure (if the cart reaches the end of the track, or the pole exceeds an angle of ± 12 degrees). At the beginning of each episode we drew the initial pole angle uniformly from $[\pm 3]$ degrees. Further, we cap episode lengths at 1000 time steps. We report on a noisy version of this problem where we add $\pm$ 50% noise to the agents actions, that is, when the agent commands a force of 1, the actual force applied is drawn uniformly from the interval $[0.5, 1.5]$. Note that, since we report on time per episode for this task, higher is better.

Mountain car: involves driving an underpowered car up a steep hill. We use the state/action space, and transition/reward dynamics as defined in [15]. In order to solve this problem the agent must first swing backwards to gain enough velocity to pass the hill. The agent receives reward -1 at each step until failure when reward 1 is received. We capped episodes in the mountain car problem to 1000 time steps. The car was initialized to a standing start (zero velocity) at a random place on the hill in each episode. Note that, since we report on time per episode for this task, lower is better.

Robot navigation: requires the agent to successfully drive a (real) robot to a specified waypoint. The state space $\mathcal{S} = (d, \theta, \dot{x}, \dot{\theta})$, where d is the distance to the goal, θ is the angle between the robot's forward direction and the line from the robot to the goal, $\dot{x}$ is the robot's forward velocity, and $\dot{\theta}$ is the robot's angular velocity. The action space is $a \in \{\ddot{x}, \ddot{\theta}\}$, which represents respective linear and angular accelerations. We restrict the accelerations to $1.0ms^{-2}$ and $1.0rads^{-2}$ with decisions made at 10Hz. This corresponds to acceleration of $1.0ms^{-1}$ per $\frac{1}{10}$ seconds for both x and θ. A reward of -1 is received at each time step. Further, a reward of 10 is received for success, -100 for leaving a 3 metre radius from the goal, and -10 for taking more than 1000 time steps; these last three events result in termination of the episode.

Table 1. The parameter setup we gave each algorithm. Here σ is the RBF tile width given to each algorithm as a fraction of the state space in each dimension after the state space was normalized to the unit hyper-sphere.

Domain	Algorithm	γ	λ	η	ξ	ϵ	σ
Mountain car	Kernel-SARSA(λ)	0.9999	0.6	0.5	—	5.0×10^{-5}	0.05
	RBF coding	0.9999	0.7	0.1	0	—	0.05
	Tile coding	0.9999	0.7	0.005	—	—	0.1
Cart pole	Kernel-SARSA(λ)	0.9999	0.7	0.1	—	5.0×10^{-7}	0.05
	RBF coding	0.9999	0.7	0.01	0.01	—	0.05
	Tile coding	0.9999	0.6	0.1	—	—	0.066
Robot navigation	Kernel-SARSA(λ)	0.9999	0.6	0.1	—	0.5	0.1
	RBF coding	0.9999	0.5	0.1	0.0	—	0.1
	Tile coding	0.9999	0.6	0.1	—	—	0.066

We used a high-fidelity simulator for training a Willow Garage PR2 robot[3] and then evaluated the learned Q-value policy on an actual PR2. In the simulator training phase, the robot's task is simply to drive directly to the waypoint. For the *in situ* robot testing phase, we gave the robot a global plan, from which it drives towards the nearest waypoint beyond a 1m radius at each time step. This has the effect of a single waypoint moving away at the same speed as the robot. At the end of the planned path, the waypoint stops moving, and the robot must drive to it. This RL problem requires a nonlinear Q-value approximation over a continuous space, and RL algorithms must deal with a high-noise environment on account of noisy actuators and an unpredictable surface response. Although the transition dynamics are noisy, we note that high-precision PR2 sensors render the state space fully observed for all practical purposes, making this an MDP.

5.2 Results

The above selection of problems demonstrate our performance in both difficult continuous state spaces requiring nonlinear approximation of Q-values and a real-world robotics navigation task.

For each MDP, we compare Kernel-SARSA(λ) to SARSA(λ) with tile coding and SARSA(λ) with RBF coding. The first metric we record for each algorithm and MDP is the memory usage in terms of the number of samples/basis functions vs. the episode number. Obviously, a small memory footprint while achieving near-optimal performance is generally desired. The second metric that we evaluate for each algorithm and MDP is the time/reward per episode, i.e., the number of steps until episode termination for *cart pole* and *mountain car* and the average reward per time step within an episode for *robot navigation*, both being recorded vs. the episode number. For the *mountain car* MDP, smaller episode length is better since episodes terminate with success, whereas for the *cart pole*

[3] `http://www.ros.org/wiki/pr2_simulator`

MDP, longer is better since episodes terminate with failure. In the *navigation task* MDP, larger average rewards per episode are better.

In the following results, each algorithm is configured with the parameter specifications in Table 1. The parameters were chosen from the following search space: $\lambda \in \{0.0, 0.1, \ldots, 0.9, 1.0\}$, $\eta \in \{1, 5\} \times 10^{-k}$, $\xi \in \{0, 1, 5\} \times 10^{-k}$, $\epsilon \in \{1, 5\} \times 10^{-k}$, $\sigma \in \frac{1}{n}$, $n \in \{5, 10, 15, 20\}$. For each algorithm and domain we chose the parameters which obtain the best result.

Memory Efficiency and Performance. Figure 1 shows the growth in the number of stored samples for Kernel-SARSA(λ), compared to the memory requirements of RBF coding and tile coding, for each of the three MDPs. We can see that Kernel-SARSA(λ) is *always* the most memory-efficient.

Figure 3 shows the time/reward for all three algorithms on all three domains. In brief, the results show that Kernel-SARSA(λ) is always among the best in terms of final episode performance (and is the best for both *cart pole* and *robot navigation*). Kernel-SARSA(λ) also learns fastest in *cart pole* while performing mid-range among the other two algorithms on both *mountain car* and *robot navigation.* This is impressive given the small amount of memory used by Kernel-SARSA(λ) relative to the other algorithms.

We now discuss results by domain in more detail:

Mountain Car: All three methods can solve this domain rather quickly. In Figure 2 (top) we see that the RBF basis functions provide the steepest decline in time needed to complete the task. Kernel-SARSA(λ) starts somewhat slower because it needs to accumulate basis functions to be able to learn the optimal policy. RBF nets and Kernel-SARSA(λ) reached an optimal policy in approximately the same number of episodes while tile coding needed many more. RBF coding showed some instabilities much later on while memory efficient Kernel-SARSA(λ) remained stable. Kernel-SARSA(λ) stores less than 150 samples, an order of magnitude smaller than best performing tile coding.

Cart Pole: As can be seen in Figure 2 (middle) Kernel-SARSA(λ) clearly outperforms both RBF nets and tile coding and learns to indefinitely balance the pole after a very small number of episodes. Neither of the comparison methods learn to reliably balance the pole during the 100 episodes. Impressively, Kernel-SARSA(λ) only stores a total of 60 samples in its representation of a near-optimal policy.

Robot Navigation: We can see that Kernel-SARSA(λ) performs the best in the long run from Figure 2 (bottom) and that for this problem, the other algorithms are far less memory efficient by at least an order of magnitude as shown in Figure 1 (bottom). While it takes a little while for Kernel-SARSA(λ) to collect an appropriate set of kernel samples before asymptoting in memory, it appears able to learn a better performing Q-function by the final episode.

Benefits of General $\boldsymbol{\lambda}$. Figure 2 shows the time/reward for Kernel-SARSA(λ) for varying values of λ on all three MDPs. The basic trend here is quite clear

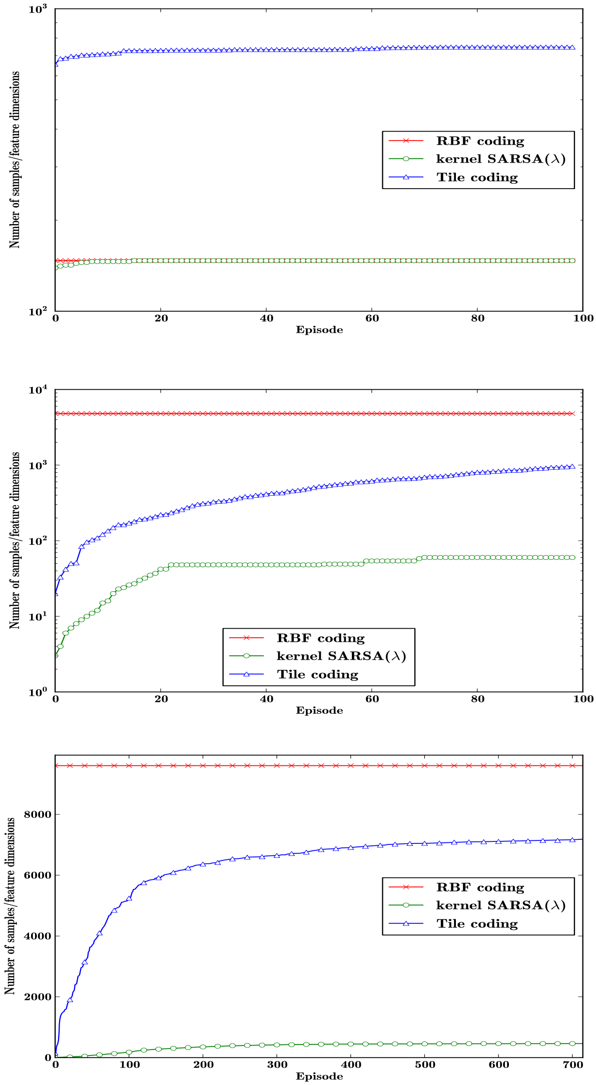


Fig. 1. Number of samples/basis functions vs. episode for all algorithms on *mountain car* (top), *cart pole* (middle), and *robot navigation* (bottom) problems. Note the vertical axis log scale on the top two plots.

— the best performing λ on all three domains satisfies $.4 \leq \lambda \leq .8$, which indicates that both for a fast initial learning rate and good asymptotic learning performance, the best $\lambda \notin \{0, 1\}$. Even further we note that $\lambda = 1$ leads to poor performance on all problems and $\lambda = 0$ leads to only mid-range performance in general.

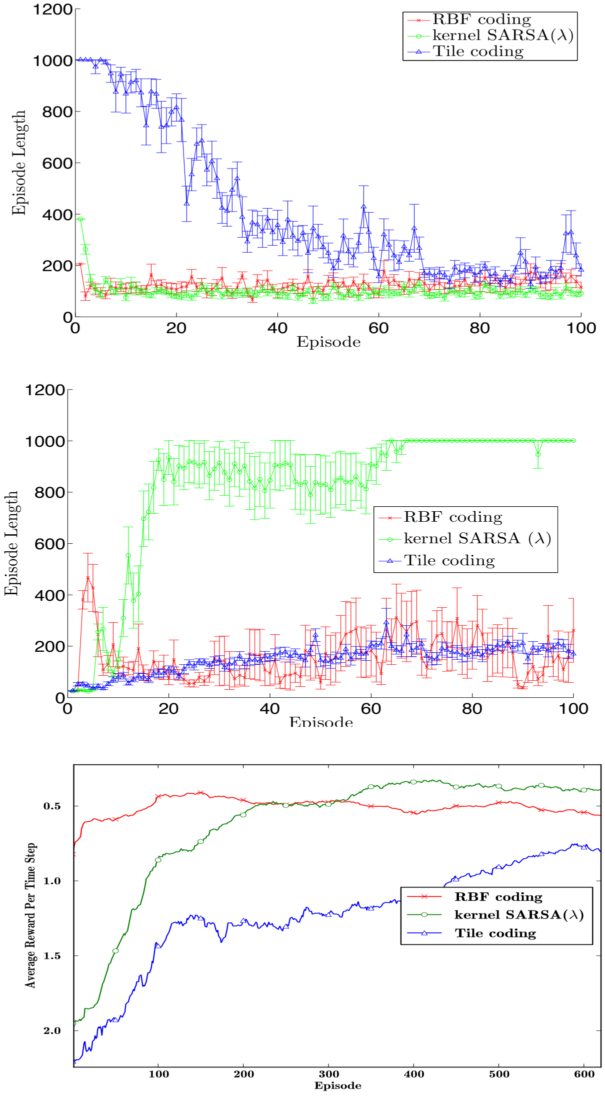


Fig. 2. Average time per episode for all algorithms and standard error over 10 runs on *mountain car* (top), *cart pole* (middle) and **moving average reward per episode** evaluated on the *robot navigation* (bottom)

Evaluation on Robot Navigation. When learning was complete in the simulator, learned Q-values were transferred to a Willow Garage PR2 robot, which was given two paths to follow as described previously. These two paths are shown in Figure 4, both demonstrating how well the agent has learned to navigate.

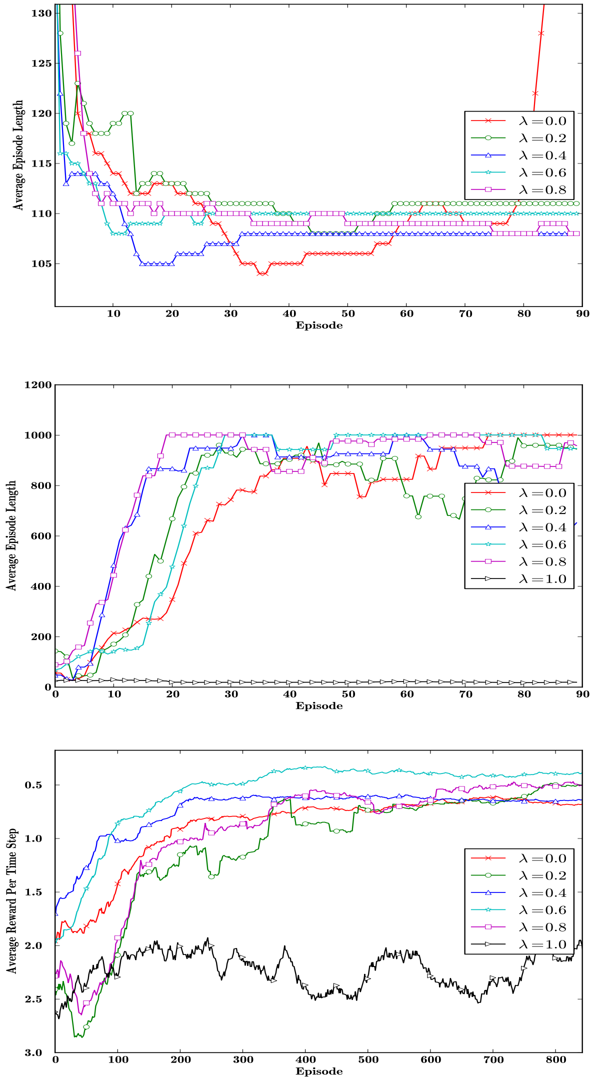


Fig. 3. Average time per episode for Kernel-SARSA(λ) with various values of λ on *mountain car* (top), *cart pole* (middle) and **moving average reward** per episode evaluated on the *robot navigation* (bottom)

We note that the more kernel samples that are stored, the more irregular the function approximation surface may be. However, Kernel-SARSA(λ)'s Projectron-based RKHS sparsification stored relatively few samples compared to other algorithms as shown in Figure 5.2, leading to a smooth Q-value approximation as exhibited in the smoothness of the navigation paths in Figure 4.

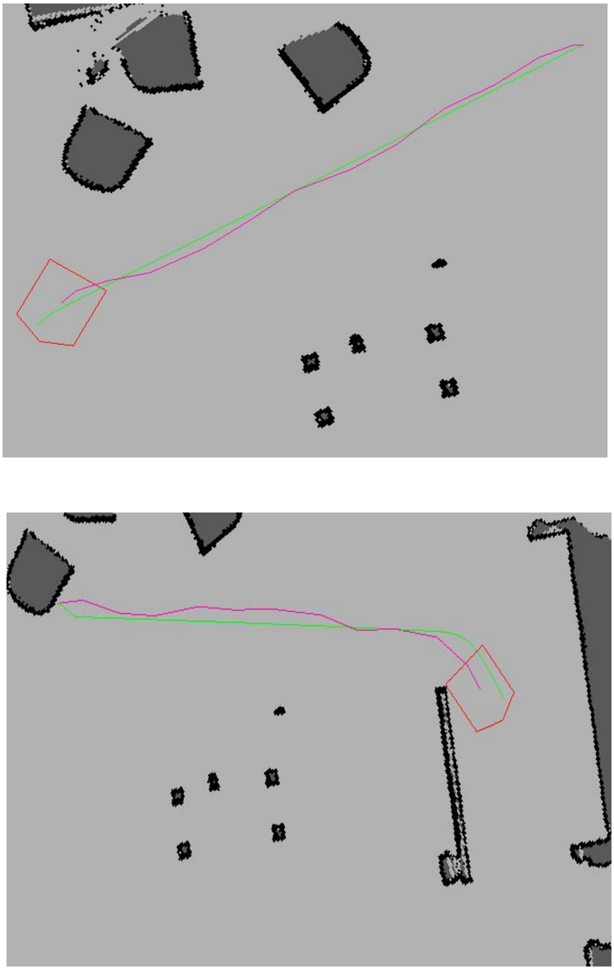


Fig. 4. The **ideal path (green), and the near-optimal path followed by Kernel-SARSA(λ) (purple)**. Here we provided the robot a virtually straight path (top) and a turn into a corridor (bottom).

6 Related Work

Both model-based and model-free approaches to using kernel methods in reinforcement learning have been proposed. In the model-based approaches, kernelized regression is used to find approximate transition and reward models which are used to obtain value function approximations. In the model-free approaches, the task of finding an approximation of the value function through regression is addressed directly as in Kernel-SARSA(λ). Gaussian Process kernel regression has been used for both approaches: [12,8] studied the model-based setting and [6,20] studied the model-free setting. [19,18] took a direct approach to replacing the inner product with a kernel in LSTD, similar to our approach in Kernel-SARSA(λ) but offline. An earlier approach at Kernel-Based Reinforcement Learning [10] that calculated a value approximation offline was modified by [7] into a model-based online approach. These approaches used kernels for "local averaging" and can be viewed as a direct approach to kernelization. Equivalence of previous kernel based approaches [20,12,18] to reinforcement learning has been proven by [16] except for the manner of regularization. But *crucially,, all of the sparse, online, model-free approaches have failed to incorporate eligibility traces for* $0 < \lambda < 1$ as we provided in the novel contribution of kernelized SARSA(λ) in this paper — the first online kernelized SARSA(λ) algorithm to show how kernelization can be extended to the eligibility trace.

7 Conclusion

We contributed the first online kernelized version of SARSA(λ) to permit arbitrary λ for $0 \leq \lambda \leq 1$ with sparsification; this was made possible via a novel kernelization of the eligibility trace maintained separately from the kernelized value function. We showed the resulting algorithm was up to an order of magnitude more memory-efficient than standard function approximation methods, while learning performance was generally on par or better. We applied Kernel-SARSA(λ) to a continuous state robotics domain and demonstrated that the learned Q-values were smoothly and accurately approximated with little memory, leading to near-optimal navigation paths on a Willow Garage PR2 robot. Importantly, we showed $.4 < \lambda < .8$ was crucial for optimal learning performance on all test problems, *emphasizing the importance of general* λ *for efficient online kernelized RL in complex, nonlinear domains* as contributed by the novel kernelization and efficient projection of the eligibility trace in Kernel-SARSA(λ) as introduced in this paper.

Acknowledgements. The first and third authors are supported by NICTA; 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. Part of this work was carried out while the first author was an intern at Willow Garage.

References

1. Aronszajn, N.: Theory of reproducing kernels. Transactions of the American Mathematical Society 68 (1950)
2. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. Springer, Heidelberg (October 2007)
3. Csato, L., Opper, M.: Sparse representation for gaussian process models. In: Advances in Neural Information Processing Systems, vol. 13 (2001)

4. Engel, Y.: Algorithms and Representations for Reinforcement Learning. PhD thesis, Hebrew University (April 2005)
5. Engel, Y., Mannor, S., Meir, R.: Sparse online greedy support vector regression. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, pp. 84–96. Springer, Heidelberg (2002)
6. Engel, Y., Mannor, S., Meir, R.: Bayes meets bellman: The gaussian process approach to temporal difference learning. In: Proc. of the 20th International Conference on Machine Learning, pp. 154–161 (2003)
7. Jong, N., Stone, P.: Kernel-based models for reinforcement learning in continuous state spaces. In: ICML Workshop on Kernel Machines and Reinforcement Learning (2006)
8. Jung, T., Stone, P.: Gaussian processes for sample efficient reinforcement learning with rmax-like exploration. In: Proceedings of the European Conference on Machine Learning (September 2010)
9. Orabona, F., Keshet, J., Caputo, B.: The projectron: a bounded kernel-based perceptron. In: ICML 2008: Proceedings of the 25th International Conference on Machine Learning, pp. 720–727. ACM, New York (2008)
10. Ormoneit, D., Sen, S.: Kernel-based reinforcement learning. Machine Learning 49, 161–178 (2002)
11. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994)
12. Rasmussen, C.E., Kuss, M.: Gaussian processes in reinforcement learning. In: Advances in Neural Information Processing Systems, vol. 16, pp. 751–759. MIT Press, Cambridge (2003)
13. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
14. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
15. Sutton, R., Barto, A.: Reinforcement Learning. The MIT Press, Cambridge (1998)
16. Taylor, G., Parr, R.: Kernelized value function approximation for reinforcement learning. In: ICML 2009: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1017–1024. ACM, New York (2009)
17. Xu, X.: A sparse kernel-based least-squares temporal difference algorithm for reinforcement learning. In: Jiao, L., Wang, L., Gao, X.-b., Liu, J., Wu, F. (eds.) ICNC 2006. LNCS, vol. 4221, pp. 47–56. Springer, Heidelberg (2006)
18. Xu, X., Hu, D., Lu, X.: Kernel-based least squares policy iteration for reinforcement learning. IEEE Transactions on Neural Networks 18(4), 973–992 (2007)
19. Xu, X., Xie, T., Hu, D., Lu, X., Xu, X., Xie, T., Hu, D., Lu, X.: Kernel least-squares temporal difference learning kernel least-squares temporal difference learning. International Journal of Information Technology, 54–63 (2005)
20. Engel, Y., Mannor, S., Meir, R.: Reinforcement learning with Gaussian processes. In: 22nd International Conference on Machine Learning (ICML 2005), Bonn, Germany, pp. 201–208 (2005)

Influence and Passivity in Social Media

Daniel M. Romero[1], Wojciech Galuba[2],
Sitaram Asur[3], and Bernardo A. Huberman[3]

[1] Cornell University, Center for Applied Mathematics, Ithaca NY, USA
[2] Distributed Information Systems Lab, EPFL, Lausanne, Switzerland
[3] Social Computing Lab, HP Labs, Palo Alto CA, USA

Abstract. The ever-increasing amount of information flowing through Social Media forces the members of these networks to compete for attention and influence by relying on other people to spread their message. A large study of information propagation within Twitter reveals that the majority of users act as passive information consumers and do not forward the content to the network. Therefore, in order for individuals to become influential they must not only obtain attention and thus be popular, but also overcome user passivity. We propose an algorithm that determines the influence and passivity of users based on their information forwarding activity. An evaluation performed with a 2.5 million user dataset shows that our influence measure is a good predictor of URL clicks, outperforming several other measures that do not explicitly take user passivity into account. We demonstrate that high popularity does not necessarily imply high influence and vice-versa.

1 Introduction

The explosive growth of Social Media has provided millions of people the opportunity to create and share content on a scale barely imaginable a few years ago. Massive participation in these social networks is reflected in the countless number of opinions, news and product reviews that are constantly posted and discussed in social sites such as Facebook, Digg and Twitter, to name a few. Given this widespread generation and consumption of content, it is natural to target one's messages to highly connected people who will propagate them further in the social network. This is particularly the case in Twitter, which is one of the fastest growing social networks on the Internet, and thus the focus of advertising companies and celebrities eager to exploit this vast new medium. As a result, ideas, opinions, and products compete with all other content for the scarce attention of the user community. In spite of the seemingly chaotic fashion with which all these interactions take place, certain topics manage to get an inordinate amount of attention, thus bubbling to the top in terms of popularity and contributing to new trends and to the public agenda of the community. How this happens in a world where crowdsourcing dominates is still an unresolved problem, but there is considerable consensus on the fact that two aspects of information transmission seem to be important in determining which content receives attention.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 18–33, 2011.

One aspect is the popularity and status of given members of these social networks, which is measured by the level of attention they receive in the form of followers who create links to their accounts to automatically receive the content they generate. The other aspect is the influence that these individuals wield, which is determined by the actual propagation of their content through the network. This influence is determined by many factors, such as the novelty and resonance of their messages with those of their followers and the quality and frequency of the content they generate. Equally important is the passivity of members of the network which provides a barrier to propagation that is often hard to overcome. Thus gaining knowledge of the identity of influential and least passive people in a network can be extremely useful from the perspectives of viral marketing, propagating one's point of view, as well as setting which topics dominate the public agenda.

In this paper, we analyze the propagation of web links on Twitter over time to understand how attention to given users and their influence is determined. We devise a general model for influence using the concept of passivity in a social network and develop an efficient algorithm similar to the HITS algorithm [14] to quantify the influence of all the users in the network. Our influence measure utilizes both the structural properties of the network as well as the diffusion behavior among users. The influence of a user thus depends not only on the size of the influenced audience, but also on their passivity. This differentiates our measure of influence from earlier ones, which were primarily based on individual statistical properties such as the number of followers or retweets [7].

We have shown through extensive evaluation that this influence model outperforms other measures of influence such as PageRank, H-index, the number of followers and the number of retweets. In addition, it has good predictive properties in that it can forecast in advance the upper bound on the number of clicks a URL can get. We have also presented case studies showing the top influential users uncovered by our algorithm. An important conclusion from the results is that the correlation between popularity and influence is quite weak, with the most influential users not necessarily being the ones with the highest popularity. Additionally, when we considered nodes with high passivity, we found the majority of them to be spammers and robot users. This demonstrates the applicability of our algorithm to automatic user categorization and filtering of online content.

2 Related Work

The study of information and influence propagation in social networks has been particularly active for a number of years in fields as disparate as sociology, communication, marketing, political science and physics. Earlier work focused on the effects that scale-free networks and the affinity of their members for certain topics had on the propagation of information [6]. Others discussed the presence of key influentials [12,11,8,5] in a social network, defined as those who are responsible for the overall information dissemination in the network. This research highlighted the value of highly connected individuals as key elements in the propagation of information through the network.

Huberman et al. [2] studied the social interactions on Twitter to reveal that the driving process for usage is a sparse hidden network underlying the friends and followers, while most of the links represent meaningless interactions. Jansen et al. [3] have examined Twitter as a mechanism for word-of-mouth advertising. They considered particular brands and products and examined the structure of the postings and the change in sentiments. Galuba et al. [4] propose a propagation model that predicts, which users will tweet about which URLs based on the history of past user activity.

There have also been earlier studies that focused on social influence and propagation. Agarwal et al. [8] have examined the problem of identifying influential bloggers in the blogosphere. They discovered that the most influential bloggers were not necessarily the most active. Aral et al [9] have distinguished the effects of homophily from influence as motivators for propagation. As to the study of influence within Twitter, Cha et al. [7] have performed a comparison of three different measures of influence - indegree, retweets and user mentions. They discovered that while retweets and mentions correlated well with each other, the indegree of users did not correlate well with the other two measures. Based on this, they hypothesized that the number of followers may not a good measure of influence. On the other hand, Weng et al [5] have proposed a topic-sensitive PageRank measure for influence in Twitter. Their measure is based on the fact that they observed high reciprocity among follower relationships in their dataset, which they attributed to homophily. However, other work [7] has shown that the reciprocity is low overall in Twitter and contradicted the assumptions of this work.

3 Twitter

3.1 Background on Twitter

Twitter is an extremely popular online microblogging service, that has gained a very large user base, consisting of more than 105 million users (as of April 2010). The Twitter graph is a directed social network, where each user chooses to follow certain other users. Each user submits periodic status updates, known as *tweets*, that consist of short messages limited in size to 140 characters. These updates typically consist of personal information about the users, news or links to content such as images, video and articles. The posts made by a user are automatically displayed on the user's profile page, as well as shown to his followers.

A *retweet* is a post originally made by one user that is forwarded by another user. Retweets are useful for propagating interesting posts and links through the Twitter community.

Twitter has attracted lots of attention from corporations for the immense potential it provides for viral marketing. Due to its huge reach, Twitter is increasingly used by news organizations to disseminate news updates, which are then filtered and commented on by the Twitter community. A number of businesses and organizations are using Twitter or similar micro-blogging services to advertise products and disseminate information to stockholders.

3.2 Dataset

Twitter provides a Search API for extracting tweets containing particular keywords. To obtain the dataset for this study, we continuously queried the Twitter Search API for a period of 300 hours starting on 10 Sep 2009 for all tweets containing the string `http`. This allowed us to acquire a complete stream of all the tweets that contain URLs. We estimated the 22 million we accumulated to be 1/15th of the entire Twitter activity at that time. From each of the accumulated tweets, we extracted the URL mentions. Each of the unique 15 million URLs in the dataset was then checked for valid formatting and the URLs shortened via the services such as `bit.ly` or `tinyurl.com` were expanded into their original form by following the HTTP redirects. For each encountered unique user ID, we queried the Twitter API for metadata about that user and in particular the user's followers and followees. The end result was a dataset of timestamped URL mentions together with the complete social graph for the users concerned.

User graph. The user graph contains those users whose tweets appeared in the stream, i.e., users that during the 300 hour observation period posted at least one public tweet containing a URL. The graph does not contain any users who do not mention any URLs in their tweets or users that have chosen to make their Twitter stream private.

For each newly encountered user ID, the list of followed users was only fetched once. Our dataset does not capture the changes occurring in the user graph over the observation period.

4 The IP Algorithm

Evidence for passivity. The users that receive information from other users may never see it or choose to ignore it. We have quantified the degree to which this occurs on Twitter (Fig. 1). An average Twitter user retweets only one in 318 URLs, which is a relatively low value. The retweeting rates vary widely across the users and the small number of the most active users play an important role in spreading the information in Twitter. This suggests that the level of user passivity should be taken into account for the information spread models to be accurate.

Assumptions. Twitter is used by many people as a tool for spreading their ideas, knowledge, or opinions to others. An interesting and important question is whether it is possible to identify those users who are very good at spreading their content, not only to those who choose to follow them, but to a larger part of the network. It is often fairly easy to obtain information about the pairwise influence relationships between users. In Twitter, for example, one can measure how much influence user A has on user B by counting the number of times B retweeted A. However, it is not very clear how to use the pairwise influence information to accurately obtain information about the relative influence each user has on the whole network. To answer this question, we design an algorithm (IP) that assigns

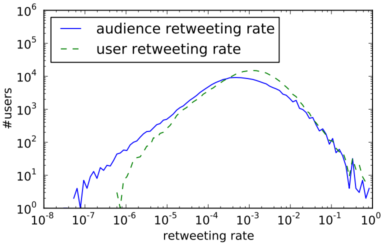


Fig. 1. Evidence for the Twitter user passivity. We measure passivity by two metrics: 1. the user retweeting rate and 2. the audience retweeting rate. The *user retweeting rate* is the ratio between the number of URLs that user i decides to retweet to the total number of URLs user i received from the followed users. The *audience retweeting rate* is the ratio between the number of user i's URLs that were retweeted by i's followers to the number of times a follower of i received a URL from i.

a relative *influence* score and a *passivity* score to every user. The passivity of a user is a measure of how difficult it is for other users to influence him. Since we found evidence that users on Twitter are generally passive, the algorithm takes into account the passivity of all the people influenced by a user, when determining the user's influence. In other words, we assume that the influence of a user depends on both the quantity and the quality of the audience she influences. In general, our model makes the following assumptions:

1. A user's influence score depends on the number of people she influences as well as their passivity.
2. A user's influence score depends on how dedicated the people she influences are. Dedication is measured by the amount of attention a user pays to a given one as compared to everyone else.
3. A user's passivity score depends on the influence of those who she's exposed to but not influenced by.
4. A user's passivity score depends on how much she rejects other user's influence compared to everyone else.

Operation. The algorithm iteratively computes both the passivity and influence scores simultaneously in the following way:

Given a weighted directed graph $G = (N, E, W)$ with nodes N, arcs E, and arc weights W, where the weights w_{ij} on arc $e = (i, j)$ represent the ratio of influence that i exerts on j to the total influence that i attempted to exert on j, the IP algorithm outputs a function $I : N \to [0, 1]$, which represents the node's relative influence on the network, and a function $P : N \to [0, 1]$ which represents the node's relative passivity.

For every arc $e = (i, j) \in E$, we define the *acceptance rate* by $u_{ij} = \dfrac{w_{i,j}}{\displaystyle\sum_{k:(k,j)\in E} w_{kj}}$.

This value represents the amount of influence that user j accepted from user i

Algorithm 1. The Influence-Passivity (IP) algorithm

```
$I_0 \leftarrow (1, 1, \ldots, 1) \in \mathbf{R}^{|N|}$;
$P_0 \leftarrow (1, 1, \ldots, 1) \in \mathbf{R}^{|N|}$;
for $i = 1$ to $m$ do
    Update $P_i$ using operation (2) and the values $I_{i-1}$;
    Update $I_i$ using operation (1) and the values $P_i$;
    for $j = 1$ to $|N|$ do
        $I_j = \frac{I_j}{\sum_{k \in N} I_k}$;
        $P_j = \frac{P_j}{\sum_{k \in N} P_k}$;
    end
end
Return $(I_m, P_m)$;
```

normalized by the total influence accepted by j from all users in the network. The acceptance rate can be viewed as the dedication or loyalty user j has to user i. On the other hand, for every $e = (j, i) \in E$ we define the *rejection rate* by $v_{ji} = \frac{1 - w_{ji}}{\sum_{k:(j,k) \in E} (1 - w_{jk})}$. Since the value $1 - w_{ji}$ is the amount of influence that user i rejected from j, then the value v_{ji} represents the influence that user i rejected from user j normalized by the total influence rejected from j by all users in the network.

The algorithm is based on the following operations:

$$I_i \leftarrow \sum_{j:(i,j) \in E} u_{ij} P_j \tag{1}$$

$$P_i \leftarrow \sum_{j:(j,i) \in E} v_{ji} I_j \tag{2}$$

Each term on the right hand side of the above operations corresponds to one of the listed assumptions. In operation 1, the term P_j corresponds to assumption 1 and the term u_{ij} corresponds to assumption 2. In operation 2, the term I_j corresponds to assumption 3 and the term v_{ji} corresponds to assumption 4. The *Influence-Passivity algorithm* (Algorithm 1) takes the graph G as the input and computes the influence and passivity for each node in m iterations.

The IP algorithm is similar to the HITS algorithm for finding authoritative web pages and hubs that link to them [14]. The passivity score corresponds to the authority score, and the influence corresponds to hub score. However, IP is different from HITS in that it operates on a weighted graph and it takes into account other properties of the network such as those referred to as "acceptance rate" and "rejection rate."

Generating the input graph. There are many ways of defining the influence graph $G = (N, E, W)$. We construct it by taking into account retweets and the follower graph in the following way: The nodes are users who tweeted at least 3 URLs. The arc (i, j) exists if user j retweeted a URL posted by user i at least once. The arc $e = (i, j)$ has weight $w_e = \frac{S_{ij}}{Q_i}$ where Q_i is the number of URLs that i mentioned and S_{ij} is the number of URLs mentioned by i and retweeted by j.

5 Evaluation

5.1 Computations

Based on the obtained dataset (§3.2) we generate the weighted graph using the method described in §4. The graph consists of approximately 450k nodes and 1 million arcs with mean weight of 0.07, and we use it to compute the PageRank, influence and passivity values for each node. The Influence-Passivity algorithm (Algorithm §1) converges to the final values in tens of iterations (Fig. 2).

PageRank. The PageRank algorithm has been widely used to rank web pages as well as people based on their authority and influence [13]. In order to compare it with the results from the IP algorithm, we compute PageRank on the weighted graph $G = (N, E, W)$ with a small change. First, since the arcs $e = (i, j) \in E$ indicate that user i exerts some influence on user j then we invert all the arcs before running PageRank on the graph while leaving the weights intact. In other words, we generate a new graph $G' = (N', E', W')$ where $N' = N$, $E' = \{(i, j) : (j, i) \in E\}$, and for each $(i, j) \in E'$ we define $w'_{ij} = w_{ji}$. This generates a new graph G' analogous to G but where the influenced users point to their influencers. Second, since the graph G' is weighted we assume that when the the random surfer of the PageRank algorithm is currently at the node i, she chooses to visit node j next with probability $\dfrac{w'_{ij}}{\displaystyle\sum_{k:(i,k)\in E'} w'_{ik}}$.

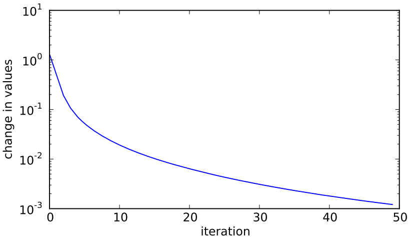


Fig. 2. IP-algorithm convergence. In each iteration we measure the sum of all the absolute changes of the computed influence and passivity values since the previous iteration.

The Hirsch Index. The Hirsch index (or H-index) is used in the scientific community in order to measure the productivity and impact of a scientist. A scientist has index h if he has published h articles which have been cited at least h times each. It has been shown that the H-index is a good indicator of whether a scientist has had high achievements such as getting the Nobel prize [16]. Analogously, in Twitter, a user has index h if h of his URL posts have been retweeted at least h times each.

5.2 Influence as a Correlate of Attention

Any measure of influence is necessarily a subjective one. However, in this case, a good measure of influence should have a high predictive power on how well the URLs mentioned by the influential users attract attention and propagate in the social network. We would expect the URLs that highly influential users propagate to attract a lot of attention and user clicks. Thus, a viable estimator of attention is the number of times a URL has been accessed.

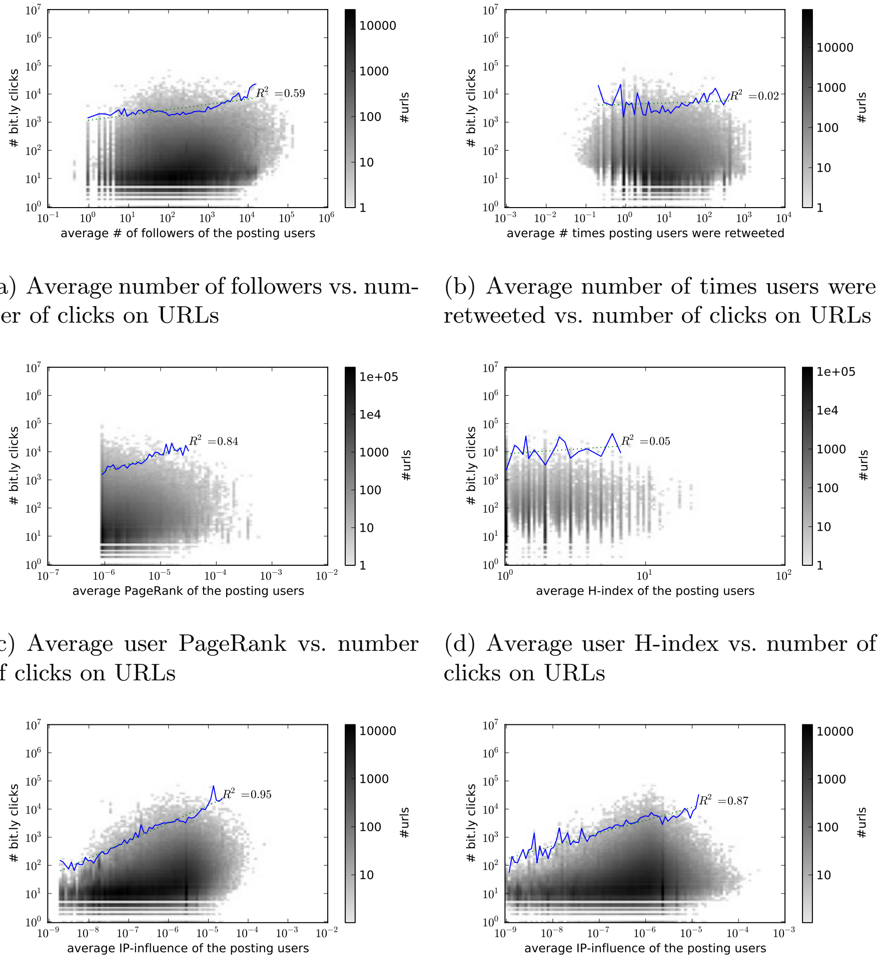
Click data. Bit.ly is a URL shortening service that for each shortened URL keeps track of how many times it has been accessed. There are 3.2M unique Bit.ly URLs in the tweets from our dataset. We have queried the Bit.ly API for the number of clicks the service has registered on each URL.

A URL my be shortened by a user who has a Bit.ly account. Each such shortening is assigned a unique per-user Bit.ly URL. To account for that we took the "global clicks" number returned by the API instead of the "user clicks" numbers. The "global clicks" number sums the clicks across all the Bit.ly shortenings of a given URL and across all the users.

URL traffic Prediction. Using the URL click data, we take several different user attributes and test how well they can predict the attention the URLs posted by the users receive (Fig. 3). It is important to note that none of the influence measures are capable of predicting the exact number of clicks. The main reason for this is that the amount of attention a URL gets is not only a function of the influence of the users mentioning it, but also of many other factors including the virality of the URL itself and more importantly, whether the URL was mentioned anywhere outside of Twitter, which is likely to be the biggest source of unpredictability in the click data.

The wide range of factors potentially affecting the Bit.ly clicks may prevent us from predicting their number accurately. However, the upper bound on that number can to a large degree be predicted. To eliminate the outlier cases, we examined how the $99.9th$ percentile of the clicks varied as the measure of influence increased.

Number of followers. The most readily available and often used by the Twitterers measure of influence is the number of followers a user has. As the Figure 3(a) shows, the number of followers of an average poster of a given URL is a relatively weak predictor of the maximum number of clicks that the URL can receive, with an R^2 value of 0.59.



(a) Average number of followers vs. number of clicks on URLs

(b) Average number of times users were retweeted vs. number of clicks on URLs

(c) Average user PageRank vs. number of clicks on URLs

(d) Average user H-index vs. number of clicks on URLs

(e) Average user IP-influence vs. number of clicks on URLs, using the retweet graph as input

(f) Average user IP-influence vs. number of clicks on URLs, using the co-mention graph as input

Fig. 3. We consider several user attributes: the number of followers, the number of times a user has been retweeted, the user's PageRank, H-index and IP-influence. For each of the 3.2M Bit.ly URLs we compute the average value of a user's attribute among all the users that mentioned that URL. This value becomes the x coordinate of the URL-point; the y coordinate is the number of clicks on the Bit.ly URL. The density of the URL-points is then plotted for each of the four user attributes. The solid line in each figure represents the $99.9th$ percentile of Bit.ly clicks at a given attribute value. The dotted line is the linear regression fit for the solid line with the fit's R^2 and slope displayed beside it.

Number of retweets. When users post URLs, their posts might be retweeted by other users. Each retweet explicitly credits the original poster of the URL (or the user from whom the retweeting user heard about the URL). The number of times a user has been credited in a retweet has been assumed to be a good measure of influence [7]. However, Figure 3(b) shows that the number of times a user has been retweeted in the past is an extremely poor predictor of the maximum number of clicks the URLs posted by that user can get.

The Hirsch Index. Figure 3(d) shows that despite the fact that in the scientific community the H-index is used as a good predictor of scientific achievements, in Twitter, it has very low correlation with URL popularity (R^2 of 0.05). This may reflect the fact that attention in the scientific community plays a symmetric role, since those who pay attention to the work of others also seek it from the same community. Thus, citations play a strategic role in the successful publishing of papers, since the expectation of authors is that referees and authors will demand attention to their work and those of their colleagues. Within Social Media such symmetry does not exist and thus the decision to forward a message to the network lacks this particularly strategic value.

PageRank. Figure 3(c) shows that the average PageRank of those who tweet a certain URL is a much better predictor of the URL's traffic than the average number of followers, retweets, or Hirsch index. The reason for the improvement could be explained by the fact that PageRank takes into account structural properties of the graph as opposed to individual measures of the users. However, figure 3(c) also shows that IP influence is a better indicator of URL popularity than PageRank. One of the main differences between the IP algorithm and PageRank is that the IP algorithm takes into account the passivity of the people a user influences and PageRank does not. This suggests that influencing users who are difficult to influence, as opposed to simply influencing many users, has a positive impact on the eventual popularity of the message that a user tweets.

IP-Influence score. As we can see in Figure 3(e), the average IP-influence of those who tweeted a certain URL can determine the maximum number of clicks that a URL will get with good accuracy, achieving an R^2 score of 0.95. Since the URL clicks are never considered by the IP algorithm to compute the user's influence, the fact that we find a very clear connection between average IP-influence and the eventual popularity of the URLs (measured by clicks) serves as an unbiased evaluation of the algorithm and demonstrates the utility of IP-influence. For example, as we can see in Figure 3(e), given a group of users having very large average IP-influence scores who post a URL we can estimate, with 99.9% certainty, that this URL will not receive more than $100,000$ clicks. On the other hand, if a group of users with very low average IP-influence score post the same URL we can estimate, with 99.9% certainty that the URL will not receive more than 100 clicks.

Furthermore, figure 4 shows that a user's IP-influence is not well correlated with the number of followers she has. This reveals interesting implications about

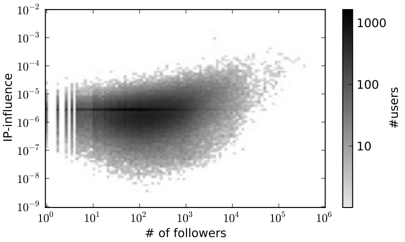


Fig. 4. For each user we place a user-point with IP-influence as the y coordinate and the x coordinate set to the number of user's followers. The density of user-points is represented in grayscale. The correlation between IP-influence and #followers is 0.44.

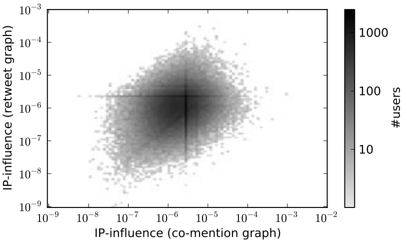


Fig. 5. The correlation between the IP-influence values computed based on two inputs: the co-mention influence graph and the retweet influence graph. The correlation between the two influence values is 0.06.

the relationship between a person's popularity and the influence she has on other people. In particular, it shows that having many followers on Twitter does not directly imply the power to influence them to click on a URL.

In the above experiments, we have used the average number of followers, retweets, PageRank, H-Index, and IP-influence of the users who posted a URL to predict the URL's traffic. We examined other choices such as using the maximum number instead of the average, and obtained similar results.

6 IP Algorithm Adaptability

As mentioned earlier (§4) there are many ways of defining a social graph in which the edges indicate pairwise influence. We have so far been using the graph based on which user retweeted which user (*retweet influence graph*). However, the explicit signals of influence such as retweets are not always available. One way of overcoming this obstacle is to use other, possibly weaker, signals of influence. In the case of Twitter, we can define an influence graph based on mentions of URLs without regard of actual retweeting in the following way.

The co-mention graph. The nodes of the *co-mention influence graph* are users who tweeted at least three URLs. The edge (i, j) exists if user j follows user i

and j mentioned at least one URL that i had previously mentioned. The edge $e = (i, j)$ has weight $w_e = \frac{S_{ij}}{F_{ij}+S_{ij}}$ where F_{ij} is the number of URLs that i mentioned and j never did and S_{ij} is the number of URLs mentioned by j and previously mentioned by i.

The resulting graph has the disadvantage that the edges are based on a much less explicit notion of influence than when based on retweets. Therefore the graph could have edges between users who do not influence each other. On the other hand, the retweeting conventions on Twitter are not uniform and therefore sometimes users who repost a URL do not necessarily credit the correct source of the URL with a retweet [15]. Hence, the influence graph based on retweets has potentially missing edges.

Since the IP algorithm has the flexibility of allowing any influence graph as input, we can compute the influence scores of the users based on the co-mention influence graph and compare with the results obtained from the retweet influence graph. As we can see in Figure 3(f), we find that the retweet graph yields influence scores that are better at predicting the maximum number of clicks a URL will obtain than the co-mention influence graph. Nevertheless, Figure 3(f) shows that the influence values obtained from the co-mention influence graph are still better at predicting URL traffic than other measures such as PageRank, number of followers, H-index or the total number of times a user has been retweeted. Furthermore, Figure 5 shows that the influence score based on both graphs do not correlate well, which suggests that considering explicit vs. implicit signals of influence can change the outcome of the IP algorithm, while at the same time maintaining its predictive value. In general, we find that the explicitness of the signal provided by the retweets yields slightly better results when it comes to predicting URL traffic, however, the influence scores based on co-mentions may surface a different set of potentially influential users.

7 Case Studies

As we mentioned earlier, one important application of the IP algorithm is ranking users by their relative influence. In this section, we present a series of rankings of Twitter users based on the influence, passivity, and number of followers.

The most influential. Table 1 shows the users with the most IP-influence in the network. We constrain the number of URLs posted to 10 to obtain this list, which is dominated by news services from politics, technology, and Social Media. These users post many links which are forwarded by other users, causing their influence to be high.

The most passive. Table 2 shows the users with the most IP-passivity in the network. Passive users are those who follow many people, but retweet a very small percentage of the information they consume. Interestingly, robot accounts (which automatically aggregate keywords or specific content from any user on the network), suspended accounts (which are likely to be spammers), and users who post extremely often are among the users with the most IP-passivity. Since

Table 1. Users with the most IP-influence (with at least 10 URLs posted in the period)

mashable	Social Media Blogger
jokoanwar	Film Director
google	Google
aplusk	Actor Ashton Kutcher
syfy	Science Fiction Channel
smashingmag	Online Developer Magazine
michellemalkin	Conservative Commentator
theonion	News Satire Organization
rww	Tech/Social Media Blogger
breakingnews	News Aggregator

Table 2. Users with the most IP-passivity

redscarebot	Keyword Aggregator
drunk_bot	Suspended
tea_robot	Keyword Aggregator
condos	Listing Aggregator
wootboot	Suspended
raybeckerman	Attorney
hashphotography	Keyword Aggregator
charlieandsandy	Suspended
ms_defy	Suspended
rpattinsonbot	Keyword Aggregator

robots "attend" to all existing tweets and only retweet certain ones, the percentage of information they forward from other users is actually very small. This explains why the IP-algorithm assigns them such high passivity scores. This also highlights a new application of the IP-algorithm: automatic identification of robot users including aggregators and spammers.

The least influential with many followers. We have demonstrated that the amount of attention a person gets may not be a good indicator of the influence they have in spreading their message. In order to make this point more explicit, we show, in Table 3, some examples of users who are followed by many people but have relatively low influence. These users are very popular and have the attention of millions of people but are not able to spread their message very far. In most cases, their messages are consumed by their followers but not considered important enough to forward to others.

The most influential with few followers. We are also able identify users with very low number of followers but high influence. Table 4 shows the users with the most influence who rank less that $100,000th$ in number of followers. We find that during the data collection period some of the users in this category ran very successful retweeting contests where users who retweeted their URLs would have the chance of winning a prize. Moreover, there is a group of users

Table 3. Users with many followers and low relative influence

User name	Category	Rank by # followers	Rank by IP-influence
thatkevinsmith	Screen Writer	33	1000
nprpolitics	Political News	41	525
eonline	TV Channel	42	1008
marthastewart	Television Host	43	1169
nba	Sports	64	1041
davidgregory	Journalist	106	3630
nfl	Sports	110	2244
cbsnews	News Channel	114	2278
jdickerson	Journalist	147	4408
newsweek	News Magazine	148	756

Table 4. Users with very few followers but high relative influence

User name	Category	Rank by # followers	Rank by IP-influence
cashcycle	Retweet Contest	153286	13
mobiliens	Retweet Contest	293455	70
jadermattos	Twitdraw	227934	134
jaum	Twitdraw	404385	143
robmillerusmc	Congressional Candidate	147803	145
sitekulite	Twitdraw	423917	149
jesse_sublett	Musician	385265	151
cyberaurora	Tech News Website	446207	163
viveraxo	Twitdraw	458279	165
fireflower_	Political Cartoons	452832	195

who post from `twitdraw.com`, a website where people can make drawings and post them on Twitter. Even though these users don't have many followers, their drawings are of very high quality and spread throughout Twitter reaching many people. Other interesting users such as local politicians and political cartoonists are also found in the list. The IP-influence measure surfaces interesting content posted by users who would otherwise be buried by popularity rankings such as number of followers.

8 Discussion

Influence as predictor of attention. As we demonstrated in §5, the IP-influence of the users is an accurate predictor of the upper bound on the total number of clicks they can get on the URLs they post. The input to the influence algorithm is a weighted graph, where the arc weights represent the influence of one user over another. This graph can be derived from the user activity in many ways, even in cases where explicit feedback in the form of retweets or "likes" is not available (§6).

Topic-based and group-based influence. The Influence-Passivity algorithm can be run on a subpgraph of the full graph or on the subset of the user activity data. For example, if only users tweeting about a certain topic are part of the graph, the IP-influence determines the most influential users in that topic. It is an open question whether the IP algorithm would be equally accurate at different graph scales.

Content ranking. The predictive power of IP-influence can be used for content filtering and ranking in order to reveal content that is most likely to receive attention based on which users mentioned that content early on. Similarly, as in the case of users, this can be computed on a per-topic or per-user-group basis.

Content filtering. We have observed from our passivity experiments that highly passive users tend to be primarily robots or spammers. This leads to an interesting extension of this work to perform content filtering, limiting the tweets to influential users and thereby reducing spam in Twitter feeds.

Influence dynamics. We have computed the influence measures over a fixed 300-hour window. However, the Social Media are a rapidly changing, real-time communication platform. There are several implications of this. First, the IP algorithm would need to be modified to take into account the tweet timestamps. Second, the IP-influence itself changes over time, which brings a number of interesting questions about the dynamics of influence and attention. In particular, whether users with spikes of IP-influence are overall more influential than users who can sustain their IP-influence over time is an open question.

9 Conclusion

Given the mushrooming popularity of Social Media, vast efforts are devoted by individuals, governments and enterprises to getting attention to their ideas, policies, products, and commentary through social networks. But the very large scale of the networks underlying Social Media makes it hard for any of these topics to get enough attention in order to rise to the most trending ones. Given this constraint, there has been a natural shift on the part of the content generators towards targeting those individuals that are perceived as influential because of their large number of followers. This study shows that the correlation between popularity and influence is weaker than it might be expected. This is a reflection of the fact that for information to propagate in a network, individuals need to forward it to the other members, thus having to actively engage rather than passively read it and rarely act on it. Moreover, since our measure of influence is not specific to Twitter it is applicable to many other social networks. This opens the possibility of discovering influential individuals within a network which can on average have a further reach than others in the same medium, regardless of their popularity.

References

1. Leskovec, J., Adamic, L.A., Huberman, B.A.: The dynamics of viral marketing. In: In Proceedings of the 7th ACM Conference on Electronic Commerce (2006)
2. Huberman, B.A., Romero, D.M., Wu, F.: Social networks that matter: Twitter under the microscope. First Monday 14(1) (January 2009)
3. Jansen, B., Zhang, M., Sobel, K., Chowdury, A.: Twitter power: Tweets as electronic word of mouth. Journal of the American Society for Information Science and Technology (2009)
4. Galuba, W., Aberer, K., Chakraborty, D., Despotovic, Z., Kellerer, W.: Outtweeting the Twitterers - Predicting Information Cascades. In: Microblogs 3rd Workshop on Online Social Networks, WOSN (2010)
5. Weng, J., Lim, E.-P., Jiang, J., He, Q.: TwitterRank: finding topic-sensitive influential twitterers. In: WSDM (2010)
6. Wu, F., Huberman, B.A., Adamic, L., Tyler, J.: Information Flow in Social Groups. Physica A 337, 327–335 (2004)
7. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, K.P.: Measuring User Influence in Twitter: The Million Follower Fallacy. In: 4th International AAAI Conference on Weblogs and Social Media, ICWSM (2010)
8. Agarwal, N., Liu, H., Tang, L., Yu, P.S.: Identifying the Influential Bloggers in a Community. In: WSDM (2008)
9. Aral, S., Muchnik, L., Sundararajan, A.: Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. Proceedings of the National Academy of Sciences 106(51), 21544–21549 (2009)
10. Watts, D.J., Dodds, P.S.: Influentials, Networks, and Public Opinion Formation. Journal of Consumer Research 34(4), 441–458 (2007)
11. Goyal, A., Bonchi, F., Lakshmanan, L.V.S.: Learning Influence Probabilitie in: Social Networks. In: WSDM (2010)
12. Domingos, P., Richardson, M.: Mining the network value of customers. In: SIGKDD (2001)
13. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30, 1–7 (1998)
14. Kleinberg, J.: Authoritative sources in a hyperlinked environment. Journal of the ACM 46(5), 604–632 (1999)
15. Danah, B., Golder, S., Lotan, G.: Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter. In: HICSS-43. IEEE, Los Alamitos (2010)
16. Hirsch, J.E.: An index to quantify an individual's scientific research output. Proceedings of the National Academy of Sciences 102(46), 16569–16572 (2005)

Preference Elicitation and Inverse Reinforcement Learning

Constantin A. Rothkopf[1] and Christos Dimitrakakis[2]

[1] Frankfurt Institute for Advanced Studies, Frankfurt, Germany
rothkfopf@fias.uni-frankfurt.de
[2] EPFL, Lausanne, Switzerland
christos.dimitrakakis@epfl.ch

Abstract. We state the problem of inverse reinforcement learning in terms of preference elicitation, resulting in a principled (Bayesian) statistical formulation. This generalises previous work on Bayesian inverse reinforcement learning and allows us to obtain a posterior distribution on the agent's preferences, policy and optionally, the obtained reward sequence, from observations. We examine the relation of the resulting approach to other statistical methods for inverse reinforcement learning via analysis and experimental results. We show that preferences can be determined accurately, even if the observed agent's policy is sub-optimal with respect to its own preferences. In that case, significantly improved policies with respect to the agent's preferences are obtained, compared to both other methods and to the performance of the demonstrated policy.

Keywords: Inverse reinforcement learning, preference elicitation, decision theory, Bayesian inference.

1 Introduction

Preference elicitation is a well-known problem in statistical decision theory [10]. The goal is to determine, whether a given decision maker prefers some events to other events, and if so, by how much. The first main assumption is that there exists a partial ordering among events, indicating relative preferences. Then the corresponding problem is to determine which events are preferred to which others. The second main assumption is the expected utility hypothesis. This posits that if we can assign a numerical *utility* to each event, such that events with larger utilities are preferred, then the decision maker's preferred choice from a set of possible *gambles* will be the gamble with the highest *expected* utility. The corresponding problem is to determine the numerical utilities for a given decision maker.

Preference elicitation is also of relevance to cognitive science and behavioural psychology, e.g. for determining rewards implicit in behaviour [19] where a proper elicitation procedure may allow one to reach more robust experimental conclusions. There are also direct practical applications, such as user modelling for determining customer preferences [3]. Finally, by analysing the apparent preferences of an expert while performing a particular task, we may be able to discover

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 34–48, 2011.

behaviours that match or even surpass the performance of the expert [1] in the very same task.

This paper uses the formal setting of preference elicitation to determine the preferences of an agent acting within a discrete-time stochastic environment. We assume that the agent obtains a sequence of (hidden to us) rewards from the environment and that its preferences have a functional form related to the rewards. We also suppose that the agent is acting nearly optimally (in a manner to be made more rigorous later) with respect to its preferences. Armed with this information, and observations from the agent's interaction with the environment, we can determine the agent's preferences and policy in a Bayesian framework. This allows us to generalise previous Bayesian approaches to inverse reinforcement learning.

In order to do so, we define a structured prior on reward functions and policies. We then derive two different Markov chain procedures for preference elicitation. The result of the inference is used to obtain policies that are significantly improved with respect to the *true preferences* of the observed agent. We show that this can be achieved even with fairly generic sampling approaches.

Numerous other inverse reinforcement learning approaches exist [1, 18, 20, 21]. Our main contribution is to provide a clear Bayesian formulation of inverse reinforcement learning as preference elicitation, with a structured prior on the agent's utilities and policies. This generalises the approach of Ramachandran and Amir [18] and paves the way to principled procedures for determining distributions on reward functions, policies and reward sequences. Performance-wise, we show that the policies obtained through our methodology easily surpass the agent's actual policy with respect to its own utility. Furthermore, we obtain policies that are significantly better than those obtained with other inverse reinforcement learning methods that we compare against.

Finally, the relation to *experimental design* for preference elicitation (see [3] for example) must be pointed out. Although this is a very interesting planning problem, in this paper we do not deal with *active* preference elicitation. We focus on the sub-problem of estimating preferences given a particular observed behaviour in a given environment and use decision theoretic formalisms to derive efficient procedures for inverse reinforcement learning.

This paper is organised as follows. The next section formalises the preference elicitation setting and relates it to inverse reinforcement learning. Section 3 presents the abstract statistical model used for estimating the agent's preferences. Section 4 describes a model and inference procedure for joint estimation of the agent's preferences and its policy. Section 5 discusses related work in more detail. Section 6 presents comparative experiments, which quantitatively examine the quality of the solutions in terms of both preference elicitation and the estimation of improved policies, concluding with a view to further extensions.

2 Formalisation of the Problem

We separate the agent's preferences (which are unknown to us) from the environment's dynamics (which we consider known). More specifically, the environment is a controlled Markov process $\nu = (\mathcal{S}, \mathcal{A}, \mathcal{T})$, with state space $\mathcal{S}$, action space $\mathcal{A}$, and transition kernel $\mathcal{T} = \{ \tau(\cdot \mid s, a) : s \in \mathcal{S}, a \in \mathcal{A} \}$, indexed in $\mathcal{S} \times \mathcal{A}$ such that $\tau(\cdot \mid s, a)$ is a probability measure[1] on $\mathcal{S}$. The dynamics of the environment are Markovian: If at time t the environment is in state $s_t \in S$ and the agent performs action $a_t \in A$, then the next state s_{t+1} is drawn with a probability independent of previous states and actions:

$$\mathbb{P}_\nu(s_{t+1} \in S \mid s^t, a^t) = \tau(S \mid s_t, a_t), \qquad S \subset \mathcal{S}, \tag{2.1}$$

where we use the convention $s^t \equiv s_1, \dots, s_t$ and $a^t \equiv a_1, \dots, a_t$ to represent sequences of variables.

In our setting, we have observed the agent acting in the environment and obtain a sequence of actions and a sequence of states:

$$D \triangleq (a^T, s^T), \qquad a^T \equiv a_1, \dots, a_T, \qquad s^T \equiv s_1, \dots, s_T.$$

The agent has an *unknown utility function*, U_t, according to which it selects actions, which we wish to discover. Here, we assume that U_t has a structure corresponding to that of reinforcement learning infinite-horizon discounted reward problems and that the agent tries to maximise the expected utility.

Assumption 1. *The agent's utility at time t is the total γ-discounted return from time t:*

$$U_t \triangleq \sum_{k=t}^{\infty} \gamma^k r_k, \tag{2.2}$$

where $\gamma \in [0, 1]$ is a discount factor, and the reward r_t is given by the (stochastic) reward function ρ so that $r_t \mid s_t = s, a_t = a \sim \rho(\cdot \mid s, a)$, $(s, a) \in \mathcal{S} \times \mathcal{A}$.

This choice establishes correspondence with the standard reinforcement learning setting.[2] The controlled Markov process and the utility define a Markov decision process [16] (MDP), denoted by $\mu = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho, \gamma)$. The agent uses some policy π to select actions with distribution $\pi(a_t \mid s_t)$, which together with the Markov decision process μ defines a Markov chain on the sequence of states, such that:

$$\mathbb{P}_{\mu,\pi}(s_{t+1} \in S \mid s^t) = \int_{\mathcal{A}} \tau(S \mid a, s_t) \, \mathrm{d}\pi(a \mid s_t), \tag{2.3}$$

where we use a subscript to denote that the probability is taken with respect to the process defined jointly by μ, π. We shall use this notational convention throughout this paper. Similarly, the *expected utility* of a policy π is denoted by $\mathbb{E}_{\mu,\pi} U_t$. We also introduce the family of Q-value functions $\{ Q^\pi_\mu : \mu \in \mathcal{M}, \pi \in \mathcal{P} \}$, where $\mathcal{M}$ is a set of MDPs, with $Q^\pi_\mu : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that:

$$Q^\pi_\mu(s, a) \triangleq \mathbb{E}_{\mu,\pi} \left(U_t \mid s_t = s, a_t = a \right). \tag{2.4}$$

[1] We assume the measurability of all sets with respect to some appropriate σ-algebra.

[2] In our framework, this is only one of the many possible assumptions regarding the form of the utility function. As an alternative example, consider an agent who collects gold coins in a maze with traps, and with a utility equal to the logarithm of the number of coins if it exists the maze, and zero otherwise.

Finally, we use Q^*_μ to denote the optimal Q-value function for an MDP μ, such that:

$$Q^*_\mu(s,a) = \sup_{\pi \in \mathcal{P}} Q^\pi_\mu(s,a), \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.5}$$

With a slight abuse of notation, we shall use Q_ρ when we only need to distinguish between different reward functions ρ, as long as the remaining components of μ remain fixed.

Loosely speaking, our problem is to estimate the reward function ρ and discount factor γ that the agent uses, given the observations s^T, a^T and some prior beliefs. As shall be seen in the sequel, this task is easier with additional assumptions on the structural form of the policy π. We derive two sampling algorithms. The first estimates a joint posterior distribution on the policy and reward function, while the second also estimates a distribution on the sequence of rewards that the agent obtains. We then show how to use those estimates in order to obtain a policy that can perform significantly better than that of the agent's original policy with respect to the agent's true preferences.

3 The Statistical Model

In the simplest version of the problem, we assume that γ, ν are known and we only estimate the reward function, given some prior over reward functions and policies. This assumption can be easily relaxed, via an additional prior on the discount factor γ and CMP ν. Let $\mathcal{R}$ be a space of reward functions ρ and $\mathcal{P}$ to be a space of policies π. We define a (prior) probability measure $\xi(\cdot \mid \nu)$ on $\mathcal{R}$ such that for any $B \subset \mathcal{R}$, $\xi(B \mid \nu)$ corresponds to our prior belief that the reward function is in B. Finally, for any reward function $\rho \in \mathcal{R}$, we define a conditional probability measure $\psi(\cdot \mid \rho, \nu)$ on the space of policies $\mathcal{P}$. Let $\rho_{\mathrm{a}}, \pi_{\mathrm{a}}$ denote the agent's true reward function and policy respectively. The joint prior on reward functions and policies is denoted by:

$$\phi(P, R \mid \nu) \triangleq \int_R \psi(P \mid \rho, \nu) \, \mathrm{d}\xi(\rho \mid \nu), \qquad P \subset \mathcal{P}, R \subset \mathcal{R}, \tag{3.1}$$

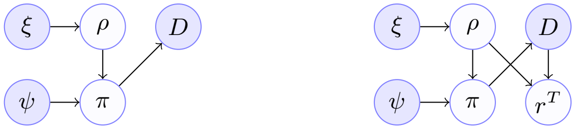
such that $\phi(\cdot \mid \nu)$ is a probability measure on $\mathcal{R} \times \mathcal{P}$. We define two models, depicted in Figure 1. The *basic model*, shown in Figure 1(a), is defined as follows:

$$\rho \sim \xi(\cdot \mid \nu), \qquad \pi \mid \rho_{\mathrm{a}} = \rho \sim \psi(\cdot \mid \rho, \nu),$$

We also introduce a *reward-augmented model*, where we explicitly model the rewards obtained by the agent, as shown in Figure 1(b):

$$\rho \sim \xi(\cdot \mid \nu), \quad \pi \mid \rho_{\mathrm{a}} = \rho \sim \psi(\cdot \mid \rho, \nu), \quad r_t \mid \rho_{\mathrm{a}} = \rho, s_t = s, a_t = a \sim \rho(\cdot \mid s, a).$$

For the moment we shall leave the exact functional form of the prior on the reward functions and the conditional prior on the policy unspecified. Nevertheless, the structure allows us to state the following:



(a) Basic model (b) Reward-augmented model

Fig. 1. Graphical model, with reward priors ξ and policy priors ψ, while ρ and π are the reward and policy, where we observe the demonstration D. Dark colours denote observed variables and light denote latent variables. The implicit dependencies on ν are omitted for clarity.

Lemma 1. *For a prior of the form specified in* (3.1)*, and given a controlled Markov process ν and observed state and action sequences s^T, a^T, where the actions are drawn from a reactive policy π, the posterior measure on reward functions is:*

$$\xi(B|s^T, a^T, \nu) = \frac{\int_B \int_{\mathcal{P}} \pi(a^T|s^T)\,\mathrm{d}\psi(\pi|\rho,\nu)\,\mathrm{d}\xi(\rho|\nu)}{\int_{\mathcal{R}} \int_{\mathcal{P}} \pi(a^T|s^T)\,\mathrm{d}\psi(\pi|\rho,\nu)\,\mathrm{d}\xi(\rho|\nu)}, \tag{3.2}$$

where $\pi(a^T \mid s^T) = \prod_{t=1}^{T} \pi(a_t|s_t)$.

Proof. Conditioning on the observations s^T, a^T via Bayes' theorem, we obtain the conditional measure:

$$\xi(B \mid s^T, a^T, \nu) = \frac{\int_B \psi(s^T, a^T \mid \rho, \nu)\,\mathrm{d}\xi(\rho \mid \nu)}{\int_{\mathcal{R}} \psi(s^T, a^T \mid \rho, \nu)\,\mathrm{d}\xi(\rho \mid \nu)}, \tag{3.3}$$

where $\psi(s^T, a^T \mid \rho, \nu) \triangleq \int_{\mathcal{P}} \mathbb{P}_{\nu,\pi}(s^T, a^T)\,\mathrm{d}\psi(\pi \mid \rho, \nu)$ is a marginal likelihood term. It is easy to see via induction that:

$$\mathbb{P}_{\nu,\pi}(s^T, a^T) = \prod_{t=1}^{T} \pi(a_t \mid s_t)\tau(s_t \mid a_{t-1}, s_{t-1}), \tag{3.4}$$

where $\tau(s_1 \mid a_0, s_0) = \tau(s_1)$ is the initial state distribution. Thus, the reward function posterior is proportional to:

$$\int_B \int_{\mathcal{P}} \prod_{t=1}^{T} \pi(a_t|s_t)\tau(s_t|a_{t-1}, s_{t-1})\,\mathrm{d}\psi(\pi|\rho,\nu)\,\mathrm{d}\xi(\rho|\nu).$$

Note that the $\tau(s_t|a_{t-1}, s_{t-1})$ terms can be taken out of the integral. Since they also appear in the denominator, the state transition terms cancel out. □

4 Estimation

While it is entirely possible to assume that the agent's policy is optimal with respect to its utility (as is done for example in [1]), our analysis can be made more interesting by assuming otherwise. One simple idea is to restrict the policy space to stationary soft-max policies:

$$\pi_\eta(a_t \mid s_t) = \frac{\exp(\eta Q^*_\mu(s_t, a_t))}{\sum_a \exp(\eta Q^*_\mu(s_t, a))}, \tag{4.1}$$

where we assumed a finite action set for simplicity. Then we can define a prior on policies, given a reward function, by specifying a prior on the inverse temperature η, such that given the reward function and η, the policy is uniquely determined.[3]

For the chosen prior (4.1), inference can be performed using standard Markov chain Monte Carlo (MCMC) methods [5]. If we can estimate the reward function well enough, we may be able to obtain policies that surpass the performance of the original policy π_{a} with respect to the agent's reward function ρ_{a}.

Algorithm 1. MH: Direct Metropolis-Hastings sampling from the joint distribution $\phi(\pi, \rho \mid a^T, s^T)$.

```
for $k = 1, \ldots$ do
    $\tilde{\rho} \sim \xi(\rho \mid \nu)$.
    $\tilde{\eta} \sim \mathcal{Gamma}(\zeta, \theta)$
    $\tilde{\pi} = \mathcal{Softmax}(\tilde{\rho}, \tilde{\eta}, \tau)$
    $\tilde{p} = \mathbb{P}_{\nu,\tilde{\pi}}(s^T, a^T)/[\xi(\rho \mid \nu) f_{\mathcal{Gamma}}(\tilde{\eta}; \zeta, \theta)]$.
    w.p. $\min\left\{1, \tilde{p}/p_{(k-1)}\right\}$ do
        $\pi_{(k)} = \tilde{\pi}$, $\eta_{(k)} = \tilde{\eta}$, $\rho_{(k)} = \tilde{\rho}$, $p_{(k)} = \tilde{p}$.
    else
        $\pi_{(k)} = \pi_{(k-1)}$, $\eta_{(k)} = \eta_{(k-1)}$, $\rho_{(k)} = \rho_{(k-1)}$, $p_{(k)} = p_{(k-1)}$.
    done
end for
```

4.1 The Basic Model: A Metropolis-Hastings Procedure

Estimation in the basic model (Fig. 1(a)) can be performed via a Metropolis-Hastings (MH) procedure. Recall that performing MH to sample from some distribution with density $f(x)$ using a proposal distribution with conditional density $g(\tilde{x} \mid x)$, has the form:

$$x_{(k+1)} = \begin{cases} \tilde{x}, & \text{w.p. } \min\left\{1, \frac{f(\tilde{x})/g(\tilde{x} \mid x_{(k)})}{f(x_{(k)})/g(x_{(k)} \mid \tilde{x})}\right\} \\ x_{(k)}, & \text{otherwise.} \end{cases}$$

In our case, $x = (\rho, \pi)$ and $f(x) = \phi(\rho, \pi \mid s^T, a^T, \nu)$.[4] We use *independent* proposals $g(x) = \phi(\rho, \pi \mid \nu)$. As $\phi(\rho, \pi \mid s^T, a^T, \nu) = \phi(s^T, a^T \mid \rho, \pi, \nu)\phi(\rho, \pi)/\phi(s^T, a^T)$, it follows that:

[3] Our framework's generality allows any functional form relating the agent's preferences and policies. As an example, we could define a prior distribution over the ϵ-optimality of the chosen policy, without limiting ourselves to soft-max forms. This would of course change the details of the estimation procedure.

[4] Here we abuse notation, using $\phi(\rho, \pi \mid \cdot)$ to denote the density or probability function with respect to a Lebesgue or counting measure associated with the probability measure $\phi(B \mid \cdot)$ on subsets of $\mathcal{R} \times \mathcal{P}$.

$$\frac{\phi(\tilde{\rho}, \tilde{\pi} \mid s^T, a^T, \nu)}{\phi(\rho, \pi \mid s^T, a^T, \nu)} = \frac{\mathbb{P}_{\nu,\tilde{\pi}}(s^T, a^T)\phi(\tilde{\rho}, \tilde{\pi} \mid \nu)}{\mathbb{P}_{\nu,\pi_{(k)}}(s^T, a^T)\phi(\rho_{(k)}, \pi_{(k)} \mid \nu)}.$$

This gives rise to the sampling procedure described in Alg. 1, which uses a gamma prior for the temperature.

4.2 The Augmented Model: A Hybrid Gibbs Procedure

The augmented model (Fig. 1(b)) enables an alternative, a two-stage hybrid Gibbs sampler, described in Alg. 2. This conditions alternatively on a reward sequence sample $r^T_{(k)}$ and on a reward function sample $\rho_{(k)}$ at the k-th iteration of the chain. Thus, we also obtain a posterior distribution on *reward sequences.*

This sampler is of particular utility when the reward function prior is conjugate to the reward distribution, in which case: (i) The reward sequence sample can be easily obtained and (ii) the reward function prior can be conditioned on the reward sequence with a simple sufficient statistic. While, sampling from the reward function posterior continues to require MH, the resulting hybrid Gibbs sampler remains a valid procedure [5], which may give better results than specifying arbitrary proposals for pure MH sampling.

As previously mentioned, the Gibbs procedure also results in a distribution over the reward sequences observed by the agent. On the one hand, this could be valuable in applications where the reward sequence is the main quantity of interest. On the other hand, this has the disadvantage of making a strong assumption about the distribution from which rewards are drawn.

Algorithm 2. G-MH: Two stage Gibbs sampler with an MH step

```
for $k = 1, \dots$ do
    $\tilde{\rho} \sim \xi(\rho \mid r^T_{(k-1)}, \nu)$.
    $\tilde{\eta} \sim \mathcal{Gamma}(\zeta, \theta)$
    $\tilde{\pi} = \mathcal{Softmax}(\tilde{\rho}, \tilde{\epsilon}, \tau)$
    $\tilde{p} = \mathbb{P}_{\nu,\tilde{\pi}}(s^T, a^T)/[\xi(\rho \mid \nu) f_{\mathcal{Gamma}}(\tilde{\eta}; \zeta, \theta)]$.
    w.p. $\min\left\{1, \tilde{p}/p_{(k-1)}\right\}$ do
        $\pi_{(k)} = \tilde{\pi}$, $\eta_{(k)} = \tilde{\eta}$, $\rho_{(k)} = \tilde{\rho}$, $p_{(k)} = \tilde{p}$.
    else
        $\pi_{(k)} = \pi_{(k-1)}$, $\eta_{(k)} = \eta_{(k-1)}$, $\rho_{(k)} = \rho_{(k-1)}$, $p_{(k)} = p_{(k-1)}$.
    done
    $r^T_{(k)} \mid s^T, a^T \sim \rho^T_{(k)}(s^T, a^T)$
end for
```

5 Related Work

5.1 Preference Elicitation in User Modelling

Preference elicitation has attracted a lot of attention in the field of user modelling and online advertising, where two main problems exist. The first is how to *model* the (uncertain) preferences of a large number of users. The second is the problem of *optimal experiment design* [see 7, ch. 14] to maximise the expected value of information through queries. Some recent models include: Braziunas and Boutilier [4] who introduced modelling of generalised additive utilities; Chu and Ghahramani [6], who proposed a Gaussian process prior over preferences, given a set of instances and pairwise relations, with applications to multiclass classification; Bonilla et al. [2], who generalised it to multiple users; [13], which proposed an additively decomposable multi-attribute utility model. Experimental design is usually performed by approximating the intractable optimal solution [3, 7].

5.2 Inverse Reinforcement Learning

As discussed in the introduction, the problems of inverse reinforcement learning and apprenticeship learning involve an agent acting in a *dynamic* environment. This makes the modelling problem different to that of user modelling where preferences are between static choices. Secondly, the goal is not only to determine the preferences of the agent, but also to find a policy that would be at least as good that of the agent with respect to the agent's own preferences.[5] Finally, the problem of experiment design does not necessarily arise, as we do not assume to have an influence over the agent's environment.

Linear Programming. One interesting solution proposed by [14] is to use a linear program in order to find a reward function that maximises the gap between the best and second best action. Although elegant, this approach suffers from some drawbacks. (a) A good estimate of the optimal policy must be given. This may be hard in cases where the demonstrating agent does not visit all of the states frequently. (b) In some pathological MDPs, there is no such gap. For example it could be that for any action a, there exists some other action a' with equal value in every state.

Policy Walk. Our framework can be seen as a generalisation of the Bayesian approach considered in [18], which does not employ a structured prior on the rewards and policies. In fact, they implicitly define the joint posterior over rewards and policies as:

$$\phi(\pi, \rho \mid s^T, a^T, \nu) = \frac{\exp\left[\eta \sum_t Q^*_\mu(s_t, a_t)\right] \xi(\rho \mid \nu)}{\phi(s^T, a^T \mid \nu)},$$

which implies that the exponential term corresponds to $\xi(s^T, a^T, \pi \mid \rho)$. This *ad hoc* choice is probably the weakest point in this approach.[6] Rearranging, we write the denominator as:

$$\xi(s^T, a^T \mid \nu) = \int_{\mathcal{R}\times\mathcal{P}} \xi(s^T, a^T \mid \pi, \rho, \nu)\, \mathrm{d}\xi(\rho, \pi \mid \nu), \tag{5.1}$$

[5] Interestingly, this can also be seen as the goal of preference elicitation when applied to multiclass classification [see 6, for example].

[6] Although, as mentioned in [18], such a choice could be justifiable through a maximum entropy argument, we note that the maximum-entropy based approach reported in [22] does not employ the value function in that way.

which is still not computable, but we can employ a Metropolis-Hastings step using $\xi(\rho \mid \nu)$ as a proposal distribution, and an acceptance probability of:

$$\frac{\xi(\pi, \rho \mid s^T, a^T)/\xi(\rho)}{\xi(\pi', \rho' \mid s^T, a^T)/\xi(\rho')} = \frac{\exp[\eta \sum_t Q^{\pi}_{\rho}(s_t, a_t)]}{\exp[\eta \sum_t Q^{\pi'}_{\rho'}(s_t, a_t)]}.$$

We note that in [18], the authors employ a different sampling procedure than a straightforward MH, called a policy grid walk. In exploratory experiments, where we examined the performance of the authors' original method [17], we have determined that MH is sufficient and that the most crucial factor for this particular method was its initialisation: as will be also be seen in Sec. 6, we only obtained a small, but consistent, improvement upon the initial reward function.

The Maximum Entropy Approach. A maximum entropy approach is reported in [22]. Given a feature function $\varPhi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^n$, and a set of trajectories $\left\{ s^{T_k}_{(k)}, a^{T_k}_{(k)} : k = 1, \ldots, n \right\}$, they obtain features $\varPhi^{T_k}_{(k)} = \left(\varPhi(s_{i,(k)}, a_{i,(k)})\right)^{T_k}_{i=1}$. They show that given empirical constraints $\mathbb{E}_{\theta,\nu} \varPhi^{T_k} = \hat{\mathbb{E}} \varPhi^{T_k}$, where $\hat{\mathbb{E}} \varPhi^{T} = \frac{1}{n} \sum_{k=1}^{n} \varPhi^{T_k}_{(k)}$ is the empirical feature expectation, one can obtain a maximum entropy distribution for actions of the form $\mathbb{P}_\theta(a_t \mid s_t) \propto e^{\theta' \varPhi(s_t, a_t)}$. If $\varPhi$ is the identity, then θ can be seen as a scaled state-action value function.

In general, maximum entropy approaches have good minimax guarantees [12]. Consequently, the estimated policy is guaranteed to be close to the agent's. However, at best, by bounding the error in the policy, one obtains a two-sided high probability bound on the relative loss. Thus, one is almost certain to perform neither much better, nor much worse that the demonstrator.

Game Theoretic Approach. An interesting game theoretic approach was suggested by [20] for apprenticeship learning. This also only requires statistics of observed features, similarly to the maximum entropy approach. The main idea is to find the solution to a game matrix with a number of rows equal to the number of possible policies, which, although large, can be solved efficiently by an exponential weighting algorithm. The method is particularly notable for being (as far as we are aware of) the only one with a high-probability upper bound on the loss relative to the demonstrating agent and no corresponding lower bound. Thus, this method may in principle lead to a significant improvement over the demonstrator. Unfortunately, as far as we are aware of, sufficient conditions for this to occur are not known at the moment. In more recent work [21], the authors have also made an interesting link between the error of a classifier trying to imitate the expert's behaviour and the performance of the imitating policy, when the demonstrator is nearly optimal.

6 Experiments

6.1 Domains

We compare the proposed algorithms on two different domains, namely on random MDPs and random maze tasks. The *Random MDP* task is a discrete-state

MDP, with four actions, such that each leads to a different, but possibly overlapping, quarter of the state set.[7] The reward function is drawn from a Beta-product hyperprior with parameters α_i and β_i, where the index i is over all state-action pairs. This defines a distribution over the parameters p_i of the Bernoulli distribution determining the probability of the agent of obtaining a reward when carrying out an action a in a particular state s.

For the *Random Maze* tasks we constructed *planar grid* mazes of different sizes, with four actions at each state, in which the agent has a probability of 0.7 to succeed with the current action and is otherwise moved to one of the adjacent states randomly. These mazes are also randomly generated, with the rewards function being drawn from the same prior. The maze structure is sampled by randomly filling a grid with walls through a product-Bernoulli distribution with parameter 1/4, and then rejecting any mazes with a number of obstacles higher than $|\mathcal{S}|/4$.

6.2 Algorithms, Priors and Parameters

We compared our methodology, using the basic (**MH**) and the augmented (**G-MH**) model, to three previous approaches. The linear programming (**LP**) based approach [14], the game-theoretic approach (**MWAL**) [20] and finally, the Bayesian inverse reinforcement learning method (**PW**) suggested in [18]. In all cases, each demonstration was a T-long trajectory s^T, a^T, provided by a demonstrator employing a softmax policy with respect to the optimal value function.

All algorithms have some parameters that must be selected. Since our methodology employs MCMC the sampling parameters must be chosen so that convergence is ensured. We found that 10^4 samples from the chain were sufficient, for both the MH and hybrid Gibbs (G-MH) sampler, with 2000 steps used as burn-in, for both tasks. In both cases, we used a gamma prior $\mathcal{Gamma}(1, 1)$ for the inverse temperature parameter η and a product-beta prior $\mathcal{Beta}^{|S|}(1, 1)$ for the reward function. Since the beta is conjugate to the Bernoulli, this is what we used for the reward sequence sampling in the G-MH sampler. Accordingly, the conditioning performed in step 11 of G-MH is closed-form.

For **PW**, we used a MH sampler seeded with the solution found by [14], as suggested by [17] and by our own preliminary experiments. Other initialisations, such as sampling from the prior, generally produced worse results. In addition, we did not find any improvement by discretising the sampling space. We also verified that the same number of samples used in our case was also sufficient for this method to converge.

The linear-programming (**LP**) based inverse reinforcement learning algorithm by Ng and Russell [14] requires the actual agent policy as input. For the *random MDP* domain, we used the maximum likelihood estimate. For the maze domain,

[7] The transition matrix of the MDPs was chosen so that the MDP was communicating (c.f. [16]) and so that each individual action from any state results in a transition to approximately a quarter of all available states (with the destination states arrival probabilities being uniformly selected and the non-destination states arrival probabilities being set to zero).

we used a Laplace-smoothed estimate (a product-Dirichlet prior with parameters equal to 1) instead, as this was more stable.

Finally, we examined the **MWAL** algorithm of Syed and Schapire [20]. This requires the cumulative discounted feature expectation as input, for appropriately defined features. Since we had discrete environments, we used the state occupancy as a feature. The feature expectations can be calculated empirically, but we obtained better performance by first computeing the transition probabilities of the Markov chain induced by the maximum likelihood (or Laplace-smoothed) policy and then calculating the expectation of these features given this chain. We set all accuracy parameters of this algorithm to 10^{-3}, which was sufficient for a robust behaviour.

6.3 Performance Measure

In order to measure performance, we plot the L_1 loss[8] of the value function of each policy relative to the optimal policy with respect to the agent's utility:

$$\ell(\pi) \triangleq \sum_{s \in \mathcal{S}} V^*_\mu(s) - V^\pi_\mu(s), \tag{6.1}$$

where $V^*_\mu(s) \triangleq \max_a Q^*_\mu(s,a)$ and $V^\pi_\mu(s) \triangleq \mathbb{E}_\pi\, Q^\pi_\mu(s,a)$.

In all cases, we average over 100 experiments on an equal number of randomly generated environments $\mu_1, \mu_2, \ldots$. For the i-th experiment, we generate a T-step-long demonstration $D_i = (s^T, a^T)$ via an agent employing a softmax policy. The same demonstration is used across all methods to reduce variance. In addition to the empirical mean of the loss, we use shaded regions to show 80% percentile across trials and error bars to display the standard error.

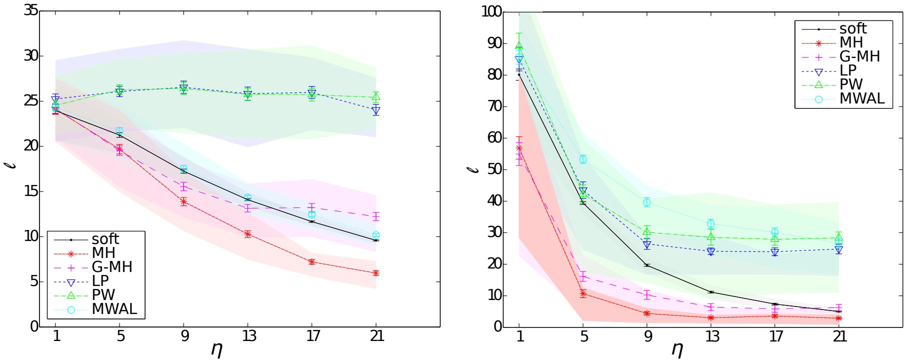
6.4 Results

We consider the loss of five different policies. The first, **soft**, is the policy of the demonstrating agent itself. The second, **MH**, is the Metropolis-Hastings procedure defined in Alg. 1, while **G-MH** is the hybrid Gibbs procedure from Alg. 2. We also consider the loss of our implementations of Linear Programming (**LP**), Policy Walk (**PW**), and **MWAL**, as summarised in Sec. 5.

We first examined the loss of greedy policies,[9] derived from the estimated reward function, as the demonstrating agent becomes greedier. Figure 2 shows results for the two different domains. It is easy to see that the **MH** sampler significantly outperforms the demonstrator, even when the latter is near-optimal. While the hybrid **Gibbs** sampler's performance lies between that of the demonstrator and the **MH** sampler, it also estimates a distribution over reward sequences as a side-effect. Thus, it could be of further value where estimation of

[8] This loss can be seen as a scaled version of the expected loss under a uniform state distribution and is a bound on the L_∞ loss. The other natural choice of the optimal policy stationary state distribution is problematic for non-ergodic MDPs.

[9] Experiments with non-greedy policies (not shown) produced generally worse results.



(a) Random MDP (b) Random Maze

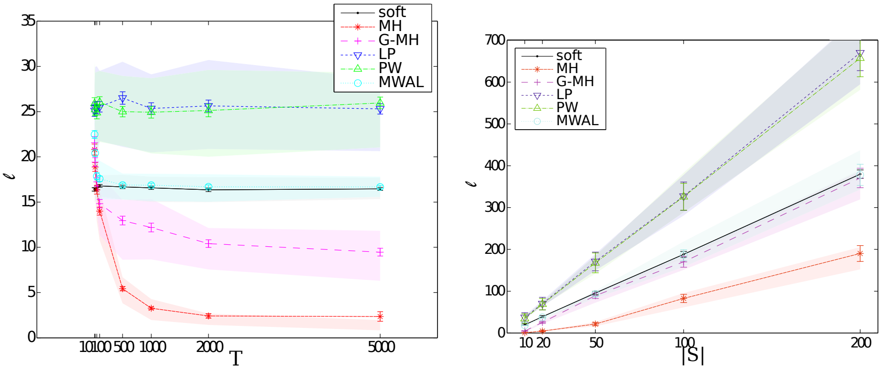
Fig. 2. Total loss ℓ with respect to the optimal policy, as a function of the inverse temperature η of the softmax policy of the demonstrator for (a) the Random MDP and (b) the Random Maze tasks, averaged over 100 runs. The shaded areas indicate the 80% percentile region, while the error bars the standard error.

reward sequences is important. We observed that the performance of the baseline methods is generally inferior, though nevertheless the **MWAL** algorithm tracks the demonstrator's performance closely.

This suboptimal performance of the baseline methods in the *Random MDP* setting cannot be attributed to poor estimation of the demonstrated policy, as can clearly be seen in Figure 3(a), which shows the loss of the greedy policy derived from each method as the amount of data increases. While the proposed samplers improve significantly as observations accumulate, this effect is smaller in the baseline methods we compared against. As a final test, we plot the relative loss in the *Random MDP* as the number of states increases in Figure 3(b). We can see that the relative performance of methods is invariant to the size of the state space for this problem.

Overall, we observed the basic model (**MH**) consistently outperforms[10] the agent in all settings. The augmented model (**G-MH**), while sometimes outperforming the demonstrator, is not as consistent. Presumably, this is due to the joint estimation of the reward sequence. Finally, the other methods under consideration on average do not improve upon the initial policy and can be, in a large number of cases, significantly worse. For the linear programming inverse RL method, perhaps this can be attributed to implicit assumptions about the MDP and the optimality of the given policy. For the policy walk inverse RL method, our belief is that its suboptimal performance is due to the very restrictive prior it uses. Finally, the performance of the game theoretic approach

[10] It was pointed out by the anonymous reviewers, that the loss we used may be biased. Indeed, a metric defined over some other state distribution, could give different rankings. However, after looking at the results carefully we determined that the policies obtained via the MH sampler were strictly dominating.



(a) Effect of amount of data

(b) Effect of environment size

Fig. 3. Total loss ℓ with respect to the optimal policy, in the Random MDP task. Figure 3(a) shows how performance improves as a function of the length T. of the demonstrated sequence. Figure 3(b) shows the effect of the number of states $|\mathcal{S}|$ of the underlying MDP. All quantities are averaged over 100 runs. The shaded areas indicate the 80% percentile region, while the error bars the standard error.

is slightly disappointing. Although it is much more robust than the other two baseline approaches, it never outperforms the demonstrator, even thought technically this is possible. One possible explanation is that since this approach is worst-case by construction, it results in overly conservative policies.

7 Discussion

We introduced a unified framework of preference elicitation and inverse reinforcement learning, presented two statistical inference models, with two corresponding sampling procedures for estimation. Our framework is flexible enough to allow using alternative priors on the form of the policy and of the agent's preferences, although that would require adjusting the sampling procedures. In experiments, we showed that for a particular choice of policy prior, closely corresponding to previous approaches, our samplers can outperform not only other well-known inverse reinforcement learning algorithms, but the demonstrating agent as well.

The simplest extension, which we have already alluded to, is the estimation of the discount factor, for which we have obtained promising results in preliminary experiments. A slightly harder generalisation occurs when the environment is not known to us. This is not due to difficulties in inference, since in many cases a posterior distribution over $\mathcal{M}$ is not hard to maintain (see for example [9, 15]). However, computing the optimal policy given a belief over MDPs is harder [9], even if we limit ourselves to stationary policies [11]. We would also like to consider more types of preference and policy priors. Firstly, the use of spatial priors for the reward function, which would be necessary for large or continuous environments. Secondly, the use of alternative priors on the demonstrator's policy.

The generality of the framework allows us to formulate different preference elicitation problems than those directly tied to reinforcement learning. For example, it is possible to estimate utilities that are not additive functions of some latent rewards. This does not appear to be easily achievable through the extension of other inverse reinforcement learning algorithms. It would be interesting to examine this in future work.

Another promising direction, which we have already investigated to some degree [8], is to extend the framework to a fully hierarchical model, with a hyperprior on reward functions. This would be particularly useful for modelling a *population* of agents. Consequently, it would have direct applications on the statistical analysis of behavioural experiments.

Finally, although in this paper we have not considered the problem of *experimental design* for preference elicitation (i.e. *active* preference elicitation), we believe is a very interesting direction. In addition, it has many applications, such as online advertising and the automated optimal design of behavioural experiments. It is our opinion that a more effective preference elicitation procedure such as the one presented in this paper is essential for the complex planning task that experimental design is. Consequently, we hope that researchers in that area will find our methods useful.

Acknowledgements. Many thanks to the anonymous reviewers for their comments and suggestions. This work was partially supported by the BMBF Project "Bernstein Fokus: Neurotechnologie Frankfurt, FKZ 01GQ0840", the EU-Project IM-CLeVeR, FP7-ICT-IP-231722, and the Marie Curie Project ESDEMUU, Grant Number 237816.

References

[1] Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the 21st International Conference on Machine Learning, ICML 2004 (2004)

[2] Bonilla, E.V., Guo, S., Sanner, S.: Gaussian process preference elicitation. In: NIPS 2010 (2010)

[3] Boutilier, C.: A POMDP formulation of preference elicitation problems. In: AAAI 2002, pp. 239–246 (2002)

[4] Braziunas, D., Boutilier, C.: Preference elicitation and generalized additive utility. In: AAAI 2006 (2006)

[5] Casella, G., Fienberg, S., Olkin, I. (eds.): Monte Carlo Statistical Methods. Springer Texts in Statistics. Springer, Heidelberg (1999)

[6] Chu, W., Ghahramani, Z.: Preference learning with gaussian processes. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 137–144. ACM, New York (2005)

[7] DeGroot, M.H.: Optimal Statistical Decisions. John Wiley & Sons, Chichester (1970)

[8] Dimitrakakis, C., Rothkopf, C.A.: Bayesian multitask inverse reinforcement learning (2011), under review

[9] Duff, M.O.: Optimal Learning Computational Procedures for Bayes-adaptive Markov Decision Processes. PhD thesis, University of Massachusetts at Amherst (2002)

[10] Friedman, M., Savage, L.J.: The expected-utility hypothesis and the measurability of utility. The Journal of Political Economy 60(6), 463 (1952)

[11] Furmston, T., Barber, D.: Variational methods for reinforcement learning. In: AISTATS, pp. 241–248 (2010)

[12] Grünwald, P.D., Philip Dawid, A.: Game theory, maximum entropy, minimum discrepancy, and robust bayesian decision theory. Annals of Statistics 32(4), 1367–1433 (2004)

[13] Guo, S., Sanner, S.: Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In: AISTATS 2010 (2010)

[14] Ng, A.Y., Russell, S.: Algorithms for inverse reinforcement learning. In: Proc. 17th International Conf. on Machine Learning, pp. 663–670. Morgan Kaufmann, San Francisco (2000)

[15] Poupart, P., Vlassis, N., Hoey, J., Regan, K.: An analytic solution to discrete Bayesian reinforcement learning. In: ICML 2006, pp. 697–704. ACM Press, New York (2006)

[16] Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, New Jersey (2005)

[17] Ramachandran, D.: Personal communication (2010)

[18] Ramachandran, D., Amir, E.: Bayesian inverse reinforcement learning. In: 20th Int. Joint Conf. Artificial Intelligence, vol. 51, pp. 2856–2591 (2007)

[19] Rothkopf, C.A.: Modular models of task based visually guided behavior. PhD thesis, Department of Brain and Cognitive Sciences, Department of Computer Science, University of Rochester (2008)

[20] Syed, U., Schapire, R.E.: A game-theoretic approach to apprenticeship learning. In: Advances in Neural Information Processing Systems, vol. 10 (2008)

[21] Syed, U., Schapire, R.E.: A reduction from apprenticeship learning to classification. In: NIPS 2010 (2010)

[22] Ziebart, B.D., Andrew Bagnell, J., Dey, A.K.: Modelling interaction via the principle of maximum causal entropy. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel (2010)

A Novel Framework for Locating Software Faults Using Latent Divergences

Shounak Roychowdhury and Sarfraz Khurshid

Department of Electrical and Computer Engineering,
University of Texas at Austin,
Austin, Texas, 78712-0240, USA
{sroychow,khursid}@ece.utexas.edu

Abstract. Fault localization, i.e., identifying erroneous lines of code in a buggy program, is a tedious process, which often requires considerable manual effort and is costly. Recent years have seen much progress in techniques for automated fault localization, specifically using program spectra – executions of failed and passed test runs provide a basis for isolating the faults. Despite the progress, fault localization in large programs remains a challenging problem, because even inspecting a small fraction of the lines of code in a large problem can require substantial manual effort. This paper presents a novel framework for fault localization based on latent divergences – an effective method for feature selection in machine learning. Our insight is that the problem of fault localization can be reduced to the problem of feature selection, where lines of code correspond to features. We also present an experimental evaluation of our framework using the Siemens suite of subject programs, which are a standard benchmark for studying fault localization techniques in software engineering. The results show that our framework enables more accurate fault localization than existing techniques.

1 Introduction

In software engineering, the process of locating the faults, i.e., selecting a set of faulty statements, in a buggy program is called *fault localization*. In machine learning, the process of selecting the most relevant features from a set of possible features is called *feature selection*. Both the processes are about selection. Therefore, the central idea of this paper is to reduce the problem of fault localization to the problem of feature selection, leverage ideas and constructs from the feature selection research to solve the fault localization problem.

There is a growing demand to seek alternatives to the traditional ways of manually debugging large-scale systems by using automated techniques. Some of the promising attempts include algorithmic debugging methods [3], static and dynamic program slicing [5], and most recently to apply data analysis techniques like Statistical Debugging [15]. Program spectrum-based methods [18], which are also known as coverage-based technique, record the execution information of a program. They deal with how statements and branches are executed with

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 49–64, 2011.

respect to a set of successful runs (positive test cases) and unsuccessful runs (negative test cases). The basic insight of the spectrum-based techniques is that similar test cases generate similar execution spectrum while dissimilar test cases generate very different types of execution spectrum. Another effective alternative technique is the predicate-based intrumentation based technique in which boolean predicates are injected within the code that collect the run-time statistics of the program. Statistical Debugging is an example of such a technique [15]. The spectrum-based methods have a two fold advantage over the predicate-based instrumentation methods: firstly it does not overly instrument the source code, and secondly a variety of code coverage tools are easily available in the market that can be used without affecting the runtime performance of source code. This paper presents a novel spectrum-based framework for fault localization based on latent divergences – an effective method for feature selection in machine learning.

The purpose of this paper is to seek a methodology in which the code coverage data is modeled as a probabilistic data source and use tools such as probabilistic divergences and their combinations to extract faulty lines of code. Our proposed method provides an alternative to methods that employ similarity based measures. For doing so, we introduce a new concept called *latent divergence*, which is in fact a product of divergences based on different conditional probabilities. These probabilties are derived from the code coverage data. Through this mechanism we extract potentially hidden information that can effectively be used for selecting faulty lines of code. Furthermore, it should be noted that latent divergence does not use any latent variables. The term “latent” in latent divergence indicates that these measures try to extract hidden information from conditional divergences.

The main contributions of this paper are as follows: 1) It proposes a novel approach of fault localization using latent divergence, a new concept based on conditional probabilistic divergences. 2) It proposes a family of measures based on latent divergence. 3) It gives a framework for using latent divergences in fault localization. 4) The experimental results show that our method performs better than state-of-the-art methods.

1.1 Related Work

In machine learning, feature selection deals with the process of selecting a subset of features without degrading accuracy or classification performance. There are three major feature selection mechanisms: filter, wrapper, and embedded selection. For details, see [11].

In recent days machine learning techniques are being applied to debugging. Neural network based methods have been proposed by Wong et al [21] in the context of fault localization. They have used classic Back-Propagation algorithm and Radial Basis Functions (RBF) based neural networks for fault localization. Jiang and Su [13] proposed Context-Aware Statistical Debugging technique that considers not only individual bug predictors and control flow paths that connect those predictors. They used Support Vector Machines (SVM) and Random Forests (RF) for statistical classification. Use of such intense computational approach makes the debugging quite slow. The authors have also observed that a single machine learning technique is not be able to properly rank predicates. These methods can be quite sensitive to way the classifiers are trained.

2 Motivation

Usually similarity measures are often used for comparing sets of data. For dichotomous data, similarity measures are based on the combination of four components of the binary contingency matrix. In the pattern recognition literature [17] several different dissimilarity measures have been widely used. The use of similarity based measures in fault localization research was started by Jones et. al [14] and Abreu et. al. [1]. The measure tuple (*suspiciousness, confidence*) was proposed by Jones et. al. in their Tarantula system. Abreu et. al. [1]. used the Ochiai metric as their similarity measure in their study. In fact, this measure was first proposed by Ochiai in a biological study [16] in 1957. While experimenting with some of these measures for our fault localization research, we observed that some of the binary similarity measures like Yule or Kulcsyzski as described in [17], which have range of $[0, \infty)$, were not effective in fault localization problem. Therefore, we were motivated to seek other alternatives to similarity-based measures for our research.

The inspiration for the proposed method comes from elementary geometry. Specifically, it comes from the theorem of the *Power of a point with respect to a circle*. This is an old and well-known result in the area of inversive geometry. See Fig.1. In simple terms, the theorem states that for a given circle with a center O and radius r, and a point A that lies outside the circle, such that P and Q are the intersections of a line through point A with the circle, then the power (p) of the point A is given by:

$$p = AP \times AQ. \tag{1}$$

This was first described by Steiner in 1826 [20]. Coxeter and Greitzer [8] showed that $p = AO^2 - r^2$. Furthermore, we use Eqn.(1)

$$AO = \sqrt{AP \times AQ + r^2}. \tag{2}$$

The interesting part is when $p = 0$ then the point A is lies on the circle. Moreover, when $p > 0$, the point lies outside the circle, and when $p < 0$ the point lies inside

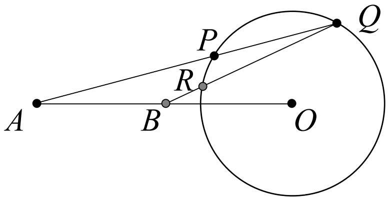


Fig. 1. Power of point with respect to a circle

the circle. Thus it might be quite effective to use this fact to compare relative positions of points.

Without any loss of generality let us consider two points A and B that lie outside the circle. In order to compute the relative distance between point A and point B it might be possible to just compare the lengths of the segments $AO = \sqrt{(AP \times AQ + r^2)}$ and $BO = \sqrt{(BR \times BQ + r^2)}$ where r is the radius the circle. To approximately compute $|AO - BO|$, it might be reasonable to ignore r for a given circle as it is a constant in both of the expressions, thus just compute $AP \times AQ$ and $BR \times BQ$ in order to find relative distance. This formed the basis of our insight to multiply two distances or two probabilistic divergences in our case to find a way of measuring *relative distance* or *separatedness* between the data points. Notice that this technique is quite different when compared to any similarity measure-based techniques. We will further explore this idea in Section 5 from the perspective of conditional probabilistic divergences.

3 An Illustrative Example

In this section we show a simple motivating example that shows the product of conditional divergences are able to accurately identify the faulty lines of code. We show a sample C code in Fig.2 and test cases in Table 1 that were used by Jones et. al. in their paper as an example. The purpose of the code is to find out the middle number. The line numbers followed a colon are shown in the left margin. Apparently a simple looking code is faulty for few inputs. We observe that the fault is in line number 7. The correct code is shown as a comment embedded in that same line.

```
#   mid() {
1:      read("Enter 3 numbers:",x,y,z);
2:      m = z;
3:      if (y<z)
4:          if (x<y)
5:             m = y;
6:          else if (x<z)
7:             m = y;   // fault1. correct: m=x
8:      else
9:         if (x>y)
10:          m = y;
11:      else if (x>z)
12:          m = x;
13:   print("Middle number is:",m);
#   }
```

Fig. 2. This simple C code is taken from Jones et al. [14] as a motivating example

Table 1. This table shows the execution-statement hit for each line using 6 different test-cases. The test cases t1-t5 generates a passed output (denoted by T(1)). The test case t6 generates a failed output (denoted by F(0)).

	1	2	3	4	5	6	7	8	9	10	11	12	13	R
t1	1	1	1	1	0	1	1	0	0	0	0	0	1	T(1)
t2	1	1	1	1	1	0	0	0	0	0	0	0	1	T(1)
t3	1	1	1	0	0	0	0	1	1	1	0	0	1	T(1)
t4	1	1	1	0	0	0	0	1	1	0	1	0	1	T(1)
t5	1	1	1	1	0	1	0	0	0	0	0	0	1	T(1)
t6	1	1	1	1	0	1	1	0	0	0	0	0	1	F(0)

Table 1 shows the execution-statement hits for 6 test cases such that t1 = (3,3,5), t2=(1,2,3), t3=(3,2,1), t4=(5,5,5), t5=(5,3,4), t6=(2,1,6). Rows denote the test cases and the columns denote the line numbers. The last column denotes output of the program with respect to the test cases. We observe that out of the 6 test cases only t6 gives an incorrect result. Fig.3 shows five subplots. Each subplot shows the values of latent divergences with respect to each line number for a set of test cases. The subplot (1) shows the result of two test cases t1 and t6. The test case t1 produces successful output (T(1)). The test case t6 produces an unsuccessful output (F(0)). The result column, which is column 14, records the result of each test case. The values of t1 and t6 are the same for all columns: from column 1 to column 13. Since there is no difference in information between

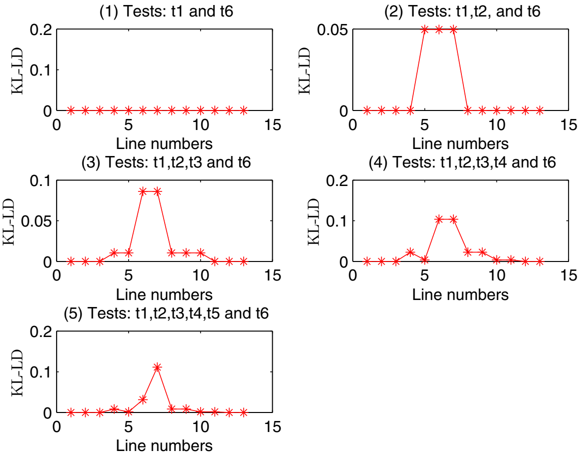


Fig. 3. This figure shows latent divergence using KL-Divergence (KL-$\mathcal{LD}$) for 6 test-cases in 5 subplots. The x-axis shows the line numbers, and y-axis shows the Latent divergence. Subplot (1) shows the latent divergence using 2 tests t1 & t6. Similarly other plots show result for different number of testcases.

two test cases, the latent divergence is zero for all the lines. The subplot (2) shows the results of adding another test case t2 to the existing set of test cases. In this scenario, we observe some changes in the values of the latent divergences at line numbers 5, 6, and 7. When we add more test cases as seen in remaining subplots, we notice that the values of latent divergence start to converge. In subplot (5), we use all the six test cases, and where we notice that, the latent divergence peaks at the line number 7 where the fault resides.

Table 2 compares the value of the metrics for Tarantula, Ochiai, and other proposed measures. Herein, we see that Tarantula and Ochiai measures have values 0.5 and 0.7071 at line 1 which is a little difficult to interpret. On careful observations, we further notice that the nonzero values of both these metrics are greater or equal to 0.5. For Tarantula, the base computed value is 0.5 $(= \frac{1}{1+1})$. For Ochiai metric, the base value is 0.7071. The computed values for the other lines are relatively close to the base value for these metrics. In this context, we believe that the power of the metric lies in its ability to differentiate the incorrect lines of code from the correct ones by significant margins. As this table shows, it is possible to localize bugs by using latent divergence.

Table 2. Comparison of Tarantula and Ochiai metrics different types of probabilistic divergences used in Definition 1 of Latent Divergence. KL-$\mathcal{LD}$ is the latent divergence uses KL-divergence, JS-$\mathcal{LD}$ uses Jensen-Shannon Divergence, R(α)-$\mathcal{LD}$ uses Renyi-Divergence, and IS-$\mathcal{LD}$ uses Itakuro-Saito divergence.

	1	2	3	4	5	6	7	8	9	10	11	12	13
Tarantula	0.50	0.50	0.50	0.625	0	0.71	0.83	0	0	0	0	0	0.5
Ochiai	0.7071	0.7071	0.7071	0.7906	0	0.8452	0.9129	0	0	0	0	0	0.7071
KL-$\mathcal{LD}$	0	0	0	0.0095	0.0037	0.0444	0.2110	0.0160	0.0160	0.0037	0.0037	0	0
JS-$\mathcal{LD}$	0	0	0	0.0366	0.0064	0.1312	0.4607	0.0366	0.0366	0.0064	0.0064	0	0
R(2)-$\mathcal{LD}$	0	0	0	0.0089	0.0015	0.0332	0.1072	0.0089	0.0089	0.0015	0.0015	0	0
R(0.5)-$\mathcal{LD}$	0	0	0	0.0019	0.0003	0.0069	0.0244	0.0019	0.0019	0.0003	0.0003	0	0
IS-$\mathcal{LD}$	0	0	0	0.8730	0.1617	2.8838	8.8456	0.8730	0.8730	0.1617	0.1617	0	0

4 Probabilistic Divergences

Probabilistic divergence is a well-known concept like dissimilarity measure but in the probabilistic space. The divergence actually measures the distance between two probability mass functions $p(x)$ and $q(x)$ such that $x \in \mathcal{X}$, and $\mathcal{X}$ is the alphabet. The standard notation for divergence is $D(p||q)$. It plays a significant role in the areas of pattern recoginition, learning and inference, and optimization.

4.1 KL-Divergence

Information and coding theory extensively use Kullback-Liebler divergence also known as KL-Divergence or Relative Entropy. Kullback-Liebler Divergence (KLD) is given by:

$$D_{KL}(p||q) = \sum_{x \in \mathcal{X}} p(x)\ log\bigg(\frac{p(x)}{q(x)}\bigg). \tag{3}$$

KL-divergence measures the distortion between two probability mass functions $p(x)$ and $q(x)$. In terms of entropies the Eqn.(3) can be written as follows: $D_{KL}(p||q) = H(p,q)$ - $H(p)$, where $H(p,q)$ is the cross-entropy between p and q and $H(p)$ is the entropy of p. It is not a true metric as it does not satisfy the symmetric and triangular properties of a metric. Jensen-Shannon Divergence (JSD) is given by the mean of two Kl-divergences:

$$D_{JS}(p||q) = \frac{1}{2}\Big(D_{KL}(p,q) + D_{KL}(q,p)\Big) \tag{4}$$

where $D_{KL}(p||q) = \sum_{x\in\mathcal{X}} p(x)\ log(\frac{p(x)}{q(x)})$ and $D_{KL}(q||p) = \sum_{x\in\mathcal{X}} q_(x)\ log(\frac{q(x)}{p(x)})$. JS-divergence satisfies three properties of being a metric.

4.2 α-Divergences

Renyi [19] proposed a following class of divergence (R(α)), which is given by:

$$D_\alpha(p||q) = \frac{1}{1-\alpha} log\Big(\sum_{x\in\mathcal{X}} \frac{p(x)^\alpha}{q(x)^{\alpha-1}}\Big). \tag{5}$$

It is a generalization of KL-divergence when $\alpha \geq 0$. When $\alpha = 0.5$, $D_{0.5}(p||q)$ $= -2log\sum_{x\in\mathcal{X}} \sqrt{p(x)q(x)}$ which is related to Bhattacharyya coefficient. When $\alpha = 2$, $D_2(p||q) = log\ E(p(x)/q(x))$ which is related to log of expected value of ratios of probabilities.

4.3 *f*-Divergences

Another broader class of divergence called $f - divergence$ has been studied by Ali-Silvey [2] and Csiszar [9]. It is given by:

$$D_f(p||q) = \sum_{x\in\mathcal{X}} q(x)\ f\Big(\frac{p(x)}{q(x)}\Big), \tag{6}$$

where f is a real valued convex function over the domain of $(0,\infty)$. This is a popular family as it can generate a variety of well-known distances like l_1 norm ($D_f(p||q) = \sum_{x\in\mathcal{X}} |p(x) - q(x)|$), squared-Hellinger distance ($D_f(p||q) = \sum_{x\in\mathcal{X}}(\sqrt{p(x)} - \sqrt{q(x)})$) etc.

4.4 Bregman-Divergences

There is another larger class of divergence called Bregman-divergence [7]. It encapsulates many well-known divergences. Like f-divergence, this class also uses convex functions to generate its members. However, the general formulation of Bregman divergence is $D_{B(f)}(p||q) = f(p) - f(q) - \langle\nabla f(q), p-q\rangle$, where f is convex function and ∇ is the gradient operator, quite different from f-divergence. The equation has flavor of first order Taylor expansion. The Euclidean distance, Mahalanobis distance, and KL Divergence are special cases of Bregman divergence. Here we will use a non-obvious member of the Bregman family in our study. Itakura-Saito Divergence (IS) [10] belongs to the class of Bregman divergence, and it is given below:

$$D_{B(f)}(p||q) = \sum_{x \in \mathcal{X}} \left(\frac{p(x)}{q(x)} - log \frac{p(x)}{q(x)} - 1 \right). \tag{7}$$

Here the convex function is $f(p) = \sum_{x \in \mathcal{X}} log(p(x))$. This divergence is quite popular in the area of speech and signal processing.

5 Latent Divergence

In this subsection, we introduce a new measure called Latent Divergence, and also propose its general family.

Let X_l denote a Bernoulli random variable for the coverage data of a particular line l such that $p_{X_l}(x) = Pr(X_l = x)$ and p_{X_l} is the probability mass function. Similarly the result is also modeled as Bernoulli random variable R and is denoted by $p_R(x) = Pr(R = x)$ and $p_R(x)$ is the probability mass function. The result of the tests passed and failed are denoted by $(T(1))$ and $(F(0))$ respectively. Since the values of X_l effect the values of R; we note that there is a joint distribution between X_l and R, and it is given by $p(x,r) = Pr(X_l = x, R = r)$. We can calculate the marginals of X_l and R as $P(X_l)$ and $P(R)$. Let us compute the conditional from the perspective of $X_l = 1$ on R, that is $p_{\{R|X_l=1\}}(x) = Pr(\{R|X_l = 1\} = x)$ which is again a conditional Bernoulli pmf. For $X_l = 0$ compute the conditional $p_{\{R|X_l=0\}}(x) = Pr(\{R|X_l = 0\} = x)$, which is again a conditional Bernoulli pmf. Using these conditional random variables we can find a certain amount of information divergence that is hidden within them when measured against random variable R. The idea is to find the amount of divergence with respect to the random variable R. That implies that we are trying to extract information like distance between $\{R|X_l = 0\}$ and R, and $\{R|X_l = 1\}$ and R respectively.

Definition 1. The latent divergence measure ($\mathcal{LD}$) between two Bernoulli random variables X_l and R is defined as follows:

$$\mathcal{LD}(X_l : R) = D(p_{\{R|X_l=1\}}||p_R)D(p_{\{R|X_l=0\}}||p_R). \tag{8}$$

Definition 2. The family of latent divergence measure is denoted by ($\mathcal{FLD}$) between two Bernoulli random variables X_l and R is defined as follows:

$$\mathcal{FLD}(X_l : R|f) = f(\mathcal{LD}(X_l : R)), \tag{9}$$

where $f : (0, \infty) \rightarrow \mathtt{R}^+$ is convex function.

Example 1. When $f_1(x) = x$, then

$$\mathcal{FLD}(X_l : R|f_1) = \mathcal{LD}(X_l : R). \tag{10}$$

Example 2. When $f_2(x) = e^x - 1$, then

$$\mathcal{FLD}_2(X_l : R|f_2) = e^{\mathcal{LD}(X_l:R)} - 1. \tag{11}$$

We show two members of the family and others can be easily be constructed. For this paper, we show only the results using the simplest latent divergence which is given by Eqn.(8).

The motivation section showed that it is possible to compare distances AO and BO using multiplicative factors. We extend that idea to compute distance like measure (separatedness) between two binary vectors with repect to a reference binary vector using probabilistic divergences. A small illustration will clarify the concept. It will also render a geometric flavor to the problem. Let there be two binary vectors such as $X_1 = (1, 0, 1, 1, 0, 1, 1)^t$ and $X_2 = (1, 0, 1, 0, 0, 0, 0)^t$ along with a reference binary vector given by $R = (1, 1, 1, 1, 0, 0, 0)^t$. After all necessary probability transformations, we plot X_1, X_2 and R in terms of their probability components p_0 and p_1 in a Cartesian space. Let p_0 be on y-axis and p_1 be on x-axis. Now see Fig. 4. Therefore, in that space we can represent X_1 as filled black circle and X_2 as filled black square. R is presented by filled red circle. The shaded circle Z_1 and shaded square Z_2 represent $Pr(\{R|X_1 = 0\} = x)$ and $Pr(\{R|X_2 = 0\} = x)$ respectively. Similarly, the unfilled circle Y_1 and unfilled square Y_2 represent $Pr(\{R|X_1 = 1\} = x)$ and $Pr(\{R|X_2 = 1\} = x)$ respectively.

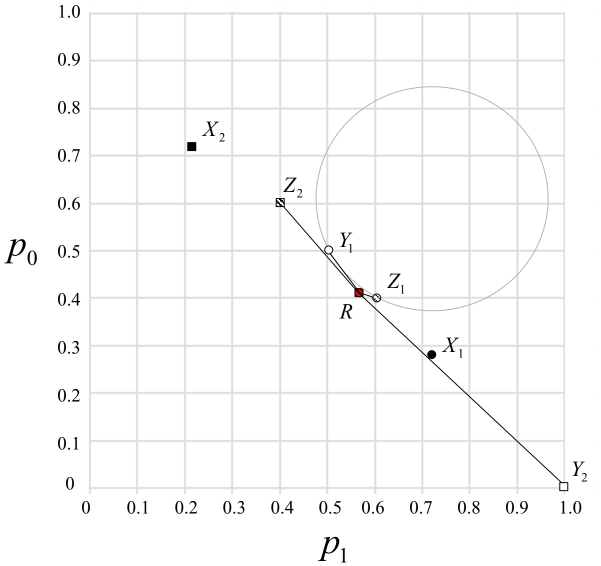


Fig. 4. A Bernoulli random variable is represented in terms components of its probabilities p_0 and p_1. The distances between points like Z_1 and R are actually divergences and should not be confused with Euclidean distance. The intention of the figure is to render a geometric idea.

We find that the latent divergence $\mathcal{LD}(X_1 : R) = 2.2262e - 005$ using KL-Divergence as a base divergence. Next for the other point, the latent divergence is $\mathcal{LD}(X_2 : R) = 0.0690$. Therefore, X_2 has more discrimination than X_1 with respect to R. It should be noted that the Fig. 4 is given to provide a geometry flavor for visualization.

The first property of the metric holds. The latent divergence $\mathcal{LD}(X_l : R) \geq 0$ because component divergences are greater than 0. The symmetric property does not hold as component divergences are not symmetric. The triangular property does not hold as divergences do not satisfy triangular property.

Theorem 1. *$\mathcal{LD}(X_l : R)$ is convex.*

Proof. We know that divergence is a convex function for all classes of divergences. Therefore,

$$D(\alpha p_1 + \beta p_2 || \alpha r_1 + \beta r_2) \leq \alpha D(p_1||r_1) + \beta D(p_2||r_2) \tag{12}$$

$$D(\alpha q_1 + \beta q_2 || \alpha r_1 + \beta r_2) \leq \alpha D(q_1||q_1) + \beta D(p_2||r_2) \tag{13}$$

Multiply Eqn.(12) and Eqn.(13), and we get the following: $D(\alpha p_1 + \beta p_2||\alpha r_1 + \beta r_2) \times D(\alpha q_1 + \beta q_2||\alpha r_1 + \beta r_2)$

$$\begin{aligned} &\leq (\alpha D(p_1||r_1) + \beta D(p_2||r_2)) \times (\alpha D(q_1||q_1) + \beta D(p_2||r_2)) \\ &= \alpha^2 D(p_1||r_1) D(q_1||r_1) + \\ &\quad \alpha\beta(D(p_2||r_2)D(q_1||r_1) + D(p_1||r_1)D(q_2||r_2)) \\ &\quad \beta^2 D(p_2||r_2) D(q_2||r_2). \end{aligned} \tag{14}$$

From the knowledge of p_1 and q_1, and along with p_2 and q_2, we find that following inequality holds true.

$$[D(\alpha p_1||\alpha r_1) - D(\alpha p_2||\alpha r_2)] \times [D(\alpha q_2||\alpha r_2) - D(\alpha q_1||\alpha r_1)] \leq 0. \tag{15}$$

Rearrange the terms of the above inequality to get

$$D(\alpha p_1||\alpha r_1)D(\alpha q_2||\alpha r_2) + D(\alpha p_2||\alpha r_2)D(\alpha q_1||\alpha r_1) \tag{16}$$

$$\leq D(\alpha p_1||\alpha r_1)D(\alpha q_1||\alpha r_1) + D(\alpha p_2||\alpha r_2)D(\alpha q_1||\alpha r_1) \tag{17}$$

Recall $\alpha + \beta = 1$. Use the above inequality in Eqn.(14) to get the following:

$$\leq \alpha^2 D(p_1||r_1)D(q_1||r_1) + \alpha\beta\big(D(\alpha p_1||\alpha r_1)D(\alpha q_1||\alpha r_1) + D(\alpha p_2||\alpha r_2)D(\alpha q_1||\alpha r_1)\big) + \beta^2 D(p_2||r_2)D(q_2||r_2) \tag{18}$$

$$= \alpha D(p_1||r_1)D(q_1||r_1) + \beta D(p_2||r_2)D(q_2||r_2). \tag{19}$$

Thus the convexity is preserved for product of two conditional probabilistic divergences. ■

Algorithm 1. Ranking Algorithm

```
Require: CodeCoverageMatrix: $X$, ResultVector: $R$
Ensure: Lines are ranked.
  $M \leftarrow$ GetNumberOfColumns($X$)
  for $l = 1$ to $M$ do
    $Y[l] \leftarrow \mathcal{LD}(X_l : R)$
  end for
  {Normalize the array $Y$ to capture the ranks; the lines having with maximum latent
  divergence will be ranked 1.}
  for $i = 1$ to $|Y|$ do
    $Z[i] \leftarrow \frac{Y[i]}{max(Y)}$
  end for
  for $i = 1$ to $|Z|$ do
    $Q[i] \leftarrow \lfloor Z[i] + M \times (1 - Z[i]) \rfloor$
  end for
```

Theorem 2. *The measures generated by latent divergence family $\mathcal{FLD}(X_l : R|f)$ are convex.*

Proof. Given that $f : \mathbb{R} \rightarrow (0, \infty)$ and is a convex function and increasing, and $g : \mathbb{R}^n \rightarrow \mathbb{R}$ and g is a convex function, then the composite function $h(x) = f \circ g$ is convex. (See [6]). Using this fact and from Theorem 1 we know $\mathcal{LD}(X_l : R)$ is convex. Thus, $\mathcal{FLD}(X_l : R|f)$=$f(\mathcal{LD}(X_l : R))$ is convex. ■

6 Latent Divergences and Fault Localization

6.1 Algorithm

Given a program P having M set of executable lines, and a set of N test cases. Firstly it is necessary to collect code coverage data X by running both successful test cases and unsuccessful test cases. The result data is collect in result vector R. Table 1 shows a sample of X matrix which has binary values and the result vector R consists of has T or F values.

After the data collection phase of X and R is over, we use the ranking algorithm described in Algorithm1. Let the l^{th} column of the code coverage matrix X be denoted $X_l = X[l]$. The ranking algorithm first computes latent divergence for each column X_l of the coverage matrix with the result vector R. It should be noted that the user selects the type of base divergence like KL-divergence, Renyi etc.

The list of latent divergence of each column is normalized to capture the rank. With respect to the algorithm the normalized data is stored in Z list. We easily observe that the list Z only contains values between 0 and 1. Furthermore, Now use the following formula in which any value of x in interval between a and b can be written as $\lambda a + (1 - \lambda)b$, a formula quite trite in the theory of convex functions. Finally the ranks of the lines are computed by using the following formula:

$$\lfloor Z[i] + M \times (1 - Z[i]) \rfloor ,$$

where M is also the number of columns of X.

Table 3. Summary of the Siemens Test Suite

Program	Description	Versions	LOC	Executable	Testcases
print_tokens	Lexical Analyser	7	565	175	4130
print_tokens2	Lexical Analyser	10	510	178	4115
replace	Pattern replacement	32	563	216	5542
schedule	Priority Scheduler	9	412	121	2650
schedule2	Priority Scheduler	10	307	112	2710
tcas	Altitude Separation	41	173	55	1608
tot-info	Information Measure	23	406	113	1052

The score measure has been used in the literature quite extensively and it is as follows:

$$score_i = \left(1 - \frac{Q[i]}{M}\right) \times 100 \tag{20}$$

We see that the score is computed for each line. Since we are only interested in the lines that are ranked number 1, unlike feature selection where the number of features is usually bigger than 1, we collect the scores reported by those lines. In other words we want the maximum value from the list of scores. On little more careful observation we may easily notice that there is a little drawback with the above equation 20 because it is highly possible that there may be more than one line (or statement) that can have the maximum suspiciousness measure at the same time. In that case the score values for all the lines that are ranked 1 will be same. However, we think that it is quite important and necessary to differentiate between the scenarios when there are multiple lines having rank 1. Thus, we introduce a weight factor W to the score value.

$$W = \frac{M + 1 - |Q^{\#1}|}{M}. \tag{21}$$

Moreover, let us denote $Q^{\#1}$ as a subset of Q whose elements are lines that are ranked 1. $|Q^{\#1}|$ is the cardinality of that subset. Thus, the score of each line then becomes:

$$score_i = \left(\left(1 - \frac{Q[i]}{M}\right) \times W\right) \times 100 \tag{22}$$

It is clear from the above equation that for when $|Q^{\#1}| = 1$, we get Eqn.(22) is equal to Eqn.(20). As only the maximum score value is a signnificant number, so the overall score is given by $score{=}max\{score_1, ..., score_M\}$.

Next we introduce another measure called Metric-Quality and it is denoted by ϕ. It actually measures a quality of a metric that is able to rank least important lines to lower ranks and most important lines to high ranks. It is defined as:

$$\phi = \sum_{i=1}^{M} \frac{1}{Q[i] \times M} \tag{23}$$

We can easily show that ϕ is bounded between $1/M$ and 1 as $Q[i]$ can vary from 1 to M. When the ϕ is small and closer to $1/M$ that implies that the metric is

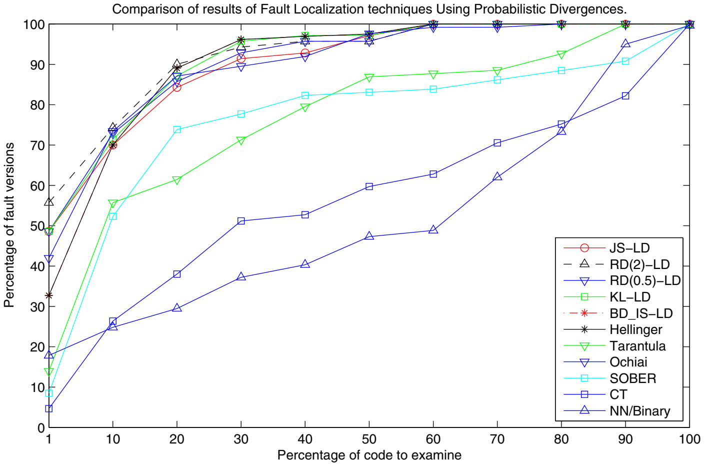


Fig. 5. The Score is plotted between percentage of code to examine and percentage of fault versions

able to suppress less important lines effectively and provide a better quality by ranking important lines much higher.

Refer to Table 2, we observe that when we use metrics like Ochiai, it generates suspiciousness values greater than to 0.9 even though the line may not be part of the bug and that is happens primarily because of the presence of a square root in Ochiai's denominator that increases its value. Such high values may confuse the programmer who might be inclined to look for statements having high suspiciousness values. Therefore, the metric's higher values might not provide the right direction for debugging.

7 Experiments

7.1 Siemens Test Suite

It is a standard practice to use the seven programs of the classic Siemens Test Suite [12] for testing feasiblities of fault localization techniques and compare them. Each program has a correct version as well as incorrect versions. All the programs in this test suite are written in C. Similar to other studies we also downloaded all the test cases from the web site [4]. Some of the test cases like version 10 of "print-tokens" and version 32 of "replace", version 9 of "Schedule2" as they were the same as the original version and therfore had no failed test cases, so they were not the part of our experiments. Similarly we also discarded some of the other test cases like version 4 and version 6 of "print_tokens" where the faults were in the header files instead of the C files. Similar observations were are mentioned in other studies as well [14]. Table 3 shows the summary of the programs. For each subject it includes the name of the program, a brief

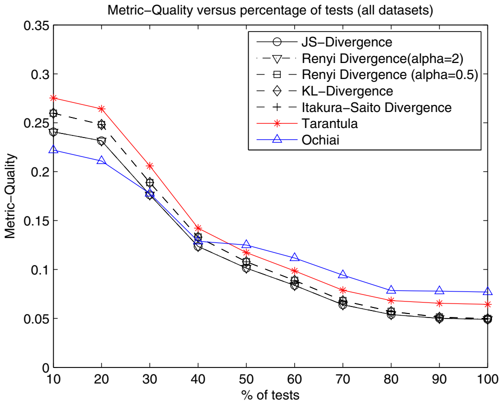


Fig. 6. The metric quality shows results for different metrics

description, the number of faulty versions, Lines of code (LOC), number of executable lines (statements), and the number of test-cases.

7.2 Results

Fig. 5 shows the plot between percentage of faulty versions versus percentage of code that needs to be examined. We find that the latent divergence performs better than other methods in general. We observe that NN/Binary [18] performs the worst while results of Tarantula and SOBER [22] are comparable. Out of the existing methods the Ochiai method seem to reasonably well. When we compare our results to the existing methods we find our method does much better Tarantula, SOBER, and CT. Even though Ochiai performs well compared to our method for only 42% of the fault versions while our method does it 33%, but just by examining 10% more code our method performs better that Ochiai. We are able to pin-point faults for 90% of the fault versions just by examining 20% of the code. We can practically cover more than 95% of the fault versions by just examining only 30% of the code the Ochiai method can only match upto 88% of the fault versions. The results obtained by using latent divergence using KL-divergence is more or less similar to all other latent divergences using different types of Renyi entropies at $\alpha = 2$ and $\alpha = 0.5$ and others.

Refer back to Eqn.(23). From the equations it is clear that the value of the ϕ should be less when most of the lines are ranked low and only few lines line are ranked high. We also observe the measure Metric-Quality (ϕ) in Fig.6 average for all programs (we also sometimes call it datasets). As shown in the plot as the number of tests are increased from 10% to 100% we can see that ϕ approaches a lower value closer to 0. The closer it is to 0, it implies that ranks of non related lines of statements are much lower and can be discard from further examination. We note latent divergence methods do better in ranking only few of the relevant lines or statements. Note that both Ochiai and Tarantula metrics

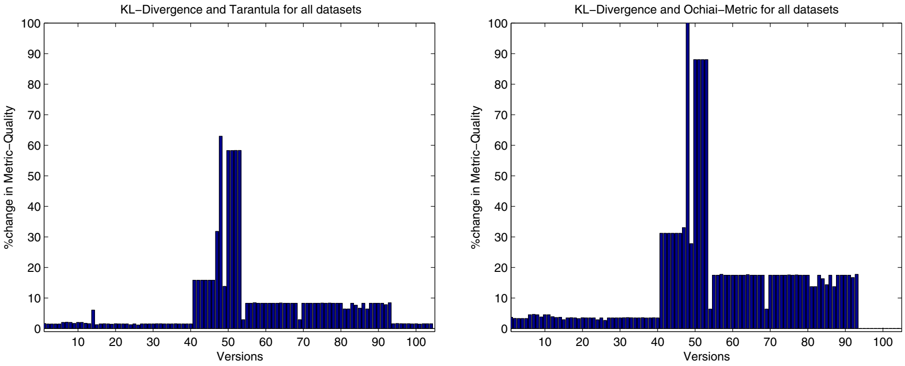


Fig. 7. (1) The percentage of the metric quality is compared between latent divergence based on KL-Divergence and Tarantula for all different program versions. (2)The percentage of the metric quality is compared between latent divergence based on KL-Divergence and Ochiai for all different program versions.

ranks of lines (statements) much higher compared to the ranks assigned by the latent divergence method on the same lines, and that may lead the programmer to wrong direction. Therefore the value of ϕ for latent divergence is naturally quite less than ϕs for the Ochiai method and the Tarantula method. Furthermore, Fig.7 shows percentage change in metric quality for each program separately. For some program versions the both Ochiai metric and Tarantula perform quite poorly. The %-change of metric-quality values shown in the Fig.7(1) and Fig.7(2) confirm trend observed in Fig.6.

8 Conclusions

In this paper we introduced a novel framework for spectrum-based fault localization using latent divergence, a novel concept based on probabilistic divergences. We show that it is feasible to use a family of latent divergences to accurately identify the lines of code that are faulty. Our experimental results show that our technique performs better than existing methods.

Acknowledgements. Khurshid's work was funded in part by the NSF under Grant Nos. IIS-0438967, and CCF-0845628, and AFOSR grant FA9550-09-1-0351.

References

1. Abreu, R., Zoeteweij, P., Golsteijn, R., van Gemund, A.J.C.: A practical evaluation of spectrum-based fault localization. Journal of Systems and Software 82(11), 1780–1792 (2009)
2. Ali, S.M., Silvey, S.D.: A General Class of Coefficients of Divergence of One Distribution from Another. J. Roy. Statist. Soc. Ser. B 28, 131–142 (1966)

3. Agrawal, H., Horgan, J.R.: Dynamic program slicing. In: Proceedings of the ACM SIGPLAN 1990 Conference on Programming Language Design and Implementation, PLDI 1990, pp. 246–256. ACM, New York (1990)
4. Aristole, http://www-static.cc.gatech.edu/aristole/tools/subjects
5. Binkley, D., Harman, M.: A survey of empirical results on program slicing. Advances in Computers 62, 106–179 (2004)
6. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (March 2004)
7. Bregman, L.M.: The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. USSR Computational Mathematics and Mathematical Physics 7(1-2), 200–217 (1967)
8. Coxeter, H.S.M., Greitzer, S.L.: Geometry Revisited (Mathematical Association of America Textbooks, 1st edn. The Mathematical Association of America (1967)
9. Csiszar, I.: On the computation of rate distortion functions. IEEE Transactions of Information Theory IT-20(1), 122–124 (1974)
10. Itakura, F., Saito, S.: An analysis-synthesis telephony based on maximum likelihood method. In: Proc. Int. Cong. Acoust., vol. c-5-5, pp. c17–c20 (1968)
11. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182 (2003)
12. Hutchins, M., Foster, H., Goradia, T., Ostrand, T.: Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In: Proceedings of the 16th International Conference on Software Engineering, ICSE 1994, pp. 191–200. IEEE Computer Society Press, Los Alamitos (1994)
13. Jiang, L., Su, Z.: Context-aware statistical debugging: from bug predictors to faulty control flow paths. In: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering, ASE 2007, pp. 184–193. ACM, New York (2007)
14. Jones, J.A., Harrold, M.J.: Empirical evaluation of the tarantula automatic fault-localization technique. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, ASE 2005, pp. 273–282. ACM, New York (2005)
15. Liblit, B., Naik, M., Zheng, A.X., Aiken, A., Jordan, M.I.: Scalable statistical bug isolation. In: Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2005, pp. 15–26. ACM, New York (2005)
16. Ochiai, A.: Zoogeographic studies on the soleoid fishes found in japan and its neighbouring regions. Bull. Jpn. Soc. Sci. Fish 22, 526–530 (1957)
17. Pekalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition: Foundations And Applications. World Scientific Publishing Co., Inc., River Edge (2005)
18. Renieris, M., Reiss, S.P.: Fault localization with nearest neighbor queries. In: International Conference on Automated Software Engineering, p. 30 (2003)
19. Renyi, A.: On measures of information and entropy. In: Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability, pp. 547–561 (1961)
20. Steiner, J.: Einige geometrische Betrachtungen. Journal für die reine und angewandte Mathematik 1, 161–184 (1826)
21. Eric Wong, W., Debroy, V., Choi, B.: A family of code coverage-based heuristics for effective fault localization. J. Syst. Softw. 83, 188–208 (2010)
22. Liu, C., Yan, X., Fei, L., Han, J., Midkiff, S.P.: SOBER: statistical model-based bug localization. In: ESEC/FSE-13: Proceedings of the 10th European Software Engineering, Lisbon, Portugal, pp. 286–295 (2005)

Transfer Learning with Adaptive Regularizers

Ulrich Rückert[1] and Marius Kloft[2,⋆]

[1] University of California, Berkeley, USA
rueckert@eecs.berkeley.edu
[2] Machine Learning Laboratory, Technische Universität Berlin, Germany
mkloft@mail.tu-berlin.de

Abstract. The success of regularized risk minimization approaches to classification with linear models depends crucially on the selection of a regularization term that matches with the learning task at hand. If the necessary domain expertise is rare or hard to formalize, it may be difficult to find a good regularizer. On the other hand, if plenty of related or similar data is available, it is a natural approach to adjust the regularizer for the new learning problem based on the characteristics of the related data. In this paper, we study the problem of obtaining good parameter values for a ℓ_2-style regularizer with feature weights. We analytically investigate a moment-based method to obtain good values and give uniform convergence bounds for the prediction error on the target learning task. An empirical study shows that the approach can improve predictive accuracy considerably in the application domain of text classification.

Keywords: transfer learning, multitask learning, regularization.

1 Introduction

Many approaches to classification optimize the sum of a data-dependent risk functional and a data-independent regularizer. Modern machine learning applications often use such methods on complex data objects, which can be described by large amounts of features. Since one has many more features than training instances in such settings, it is important to choose good regularization. Ideally, one would want to choose a regularizer that matches well with the unknown data-generating distribution. Finding such a good regularizer can either be done based on the available data (which might lead to overfitting) or based on domain expertise or meta knowledge, which is often rare or requires significant amount of work. Modern automated data processing systems, on the other hand, have led to the availability of vast amounts of potentially related data, which might help in selecting a good regularizer.

In this paper we address the problem of automatically adapting the regularizer for a *target learning problem*, if one has access to a (possibly large) number of related *source learning tasks*. To do so, we choose a highly parameterized regularizer for the target learning problem and try to obtain good settings for

[⋆] A part of the work was fone while MK was with University of California, Berkeley.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 65–80, 2011.

the parameters from the source data sets. We frame the problem theoretically using a frequentist hierarchical model, similar to the ones by Baxter [4] and Ando and Zhang [1]. However, instead of bounding the average prediction error over all learning tasks (*multitask learning*), we give bounds only for the prediction error on the target task (*inductive transfer*). We also do not make any fixed assumption about how the source and target learning tasks are related, such as transformation-based relatedness [5,12] or preprocessing-based relatedness [11,17]. Instead, we start from a worst-case analysis and only make the assumption that source and target learning tasks are drawn i.i.d. from a fixed but unknown distribution. We then show how one can add additional assumptions to improve those worst case bounds in particular situations. The resulting uniform convergence bounds relate the success of the regularization parameters obtained from the source data sets to the number of source data sets and quantifies the trade-off between estimation and approximation errors.

We evaluate our approach empirically in the application domains of text classification and predicting molecular structure-activity relationships. The results indicate that our approach works at least as well as a regular SVM and in a few cases yields drastic gains in prediction accuracy up to 19% over approaches that do not transfer information from the source tasks.

Our main contributions can be summarized as follows:

- We present a novel approach to transfer learning, for which we show upper bounds on the generalization error on the target task in a hierarchical i.i.d. setup.
- We show how our bound can be further tightened when distribution-dependent information is available. We demonstrate that the so-obtained generalization bounds can be strictly tighter than standard results.
- We show that our approach works well in the domain of text classification, yielding gains in accuracy of up to 19% compared to a regular SVM and the approach of Evgeniou & Pontil [8].

Finally, we would like to mention that our method is easy-to-use since one just needs a regular SVM implementation. We thus believe that our method could be useful to other researchers for exploring new application domains in which transfer learning might be helpful. Our implementation will be made available with the final version of this paper.

2 Regularization Adaptation with Transfer Learning

Let us now describe the setting more formally. We are given a space of data objects $\mathcal{X}$ that are embedded in an Euclidean feature space, i.e. $\mathcal{X} \simeq \mathbb{R}^m$, and a set of binary class labels $\mathcal{Y} = \{-1, +1\}$. We assume that nature poses a sequence of source learning tasks $T^1, \ldots, T^p$ and one target learning task T°. We assume that all these learning tasks are drawn i.i.d. from a fixed but unknown distribution $\mathcal{T}$. The goal is to find a good classifier for the target learning problem T°. For each learning task $T^i = (X^i, Y^i)$ we are given a sample of training data

$X^i = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_{n^i}\}$, and labels $Y^i = \{y_1, ..., y_{n^i}\}$, drawn i.i.d. from some unknown distribution, which in turn is drawn from $\mathcal{T}$. For ease of notation, we assume $n := n^1 = \ldots = n^p$ in the following.

As we are not interested in the actual data-generating distributions for the source tasks, but only the distribution of the observed data, we will not distinguish between "true" and "empirical" source distributions. Instead we simply assume that each source task distribution P^i is defined with regard to the sample (X^i, Y^i). This means we write $\Pr_{P^i}$ to denote the sample probability measure $\Pr_{P^i}(x, y) := \frac{1}{n^i} \sum_{(x^i, y^i) \in (X^i, Y^i)} I[x = x^i \wedge y = y^i]$ and $\mathbb{E}_{(x,y) \sim P^i}$ for the sample expectation $\mathbb{E}_{(x,y) \sim P^i}[f(x, y)] := \frac{1}{n^i} \sum_{(x^i, y^i) \in (X^i, Y^i)} f(x^i, y^i)$. For the target learning task T°, we follow the same convention, but assume that we have seen only a smaller fraction $n^\circ << n$ of all target examples. This means that the "true" probabilities $\Pr_{P^\circ}$ and expectations $\mathbb{E}_{P^\circ}$ are still defined with regard to the sample $X^\circ = \{x_1^\circ, \ldots, x_n^\circ\}$. However, for learning a classifier, we only have access to a smaller subset $\{x_1^\circ, \ldots, x_{n^\circ}^\circ\} \subset X^\circ$ of examples.

We denote by $\mathbb{E}$ and Pr the overall expectation and probability over the choice of both the learning tasks and a particular training sample, unless stated otherwise, while conditional expectations will be marked by a subscript; for example, $\mathbb{E}_{(\boldsymbol{x},y) \sim P^\circ}$ takes the expectation over the target data generating distribution P°, but is still a random variable with regard to the learning task generating distribution $\mathcal{T}$. In the cases where we take the overall probability, the random quantities usually only depend on the drawing of the target task P° from the data set generating distribution $\mathcal{T}$. Therefore, it is usually safe to assume $\mathbb{E} = \mathbb{E}_{\mathcal{T}}$ and $\Pr = \Pr_{\mathcal{T}}$ in the following.

For the source data sets $(X^1, Y^1), \ldots, (X^p, Y^p)$, we would like to find linear classifiers $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^p \in \mathbb{R}^{m^i}$ whose loss $\mathrm{er}^i(\boldsymbol{w}^i)$ is as small as possible, while constraining $\|\boldsymbol{w}\|_2 \leq C$. For ease of notation, we set $C = 1$, but the following results also hold for other choices of C. Define

$$\forall i = 1, ..., p : \quad \boldsymbol{w}^i := \operatorname*{argmin}_{\boldsymbol{w} : \|\boldsymbol{w}\|_2 \leq 1} \ \mathrm{er}^i(\boldsymbol{w}), \tag{1}$$

$$\text{where } \ \forall \boldsymbol{w} : \ \mathrm{er}^i(\boldsymbol{w}) := \frac{1}{n} \sum_{j=1}^{n} \ell(\boldsymbol{w}^\top \boldsymbol{x}_j^i y_j^i).$$

Here, $\ell : \mathbb{R} \to [0, 1]$ is a loss function measuring the quality of a prediction. Suppose the criterion has a unique solution $\boldsymbol{w}^i$. We can then view the $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^p$ as a random sample of empirical risk minimizers.

However, our goal is to find a vector $\boldsymbol{w}$ that minimizes the expected error $\mathrm{er}^\circ(\boldsymbol{w})$ on the *target* task, while we can only observe the empirical error $\hat{\mathrm{er}}^\circ(\boldsymbol{w})$:

$$\forall \boldsymbol{w} : \quad \mathrm{er}^\circ(\boldsymbol{w}) := \mathop{\mathbb{E}}_{(\boldsymbol{x},y) \sim P^\circ} \ell(\boldsymbol{w}^\top \boldsymbol{x} y) = \frac{1}{n} \sum_{j=1}^{n} \ell(\boldsymbol{w}^\top \boldsymbol{x}_j^\circ y_j^\circ), \tag{2}$$

$$\hat{\mathrm{er}}^\circ(\boldsymbol{w}) := \frac{1}{n^\circ} \sum_{j=1}^{n^\circ} \ell(\boldsymbol{w}^\top \boldsymbol{x}_j^\circ y_j^\circ).$$

To this aim, we could employ a standard approach such as (1). However, since we know that T° is drawn from the same distribution $\mathcal{T}$ as the $T^1,\ldots,T^p$, it would make sense to re-use some of the information in the source data sets for the selection of the target classifier. In the following we do so by using an adjustable regularization term, which is modeled on base of the observed source tasks:

Proposed Transfer Learning Approach

$$\hat{\boldsymbol{w}}^\circ := \operatorname*{argmin}_{\boldsymbol{w}\in B_b} \hat{\text{er}}^\circ(\boldsymbol{w}), \tag{T}$$

where B_b is a regularizer depending on the source tasks. The main idea is that the new regularizer forces the classifier to be from a more restricted set B_b, whose size and form depends on the source learning tasks $T^1,\ldots,T^p$ and a scale parameter $b \in \mathbb{R}$.

More specifically, the B_b is designed to keep the favorable properties of ℓ_2 regularization, but to transfer information about the relevance of individual features or feature groups. A feature, which gets assigned large weights on most source data sets is likely to also be informative on the target data set. Thus, it makes sense to adjust the regularizer for the target learning task so that it encourages the assignment of large weights to informative features and to penalize the assignment of large weights to features which have not received considerable weights on the source learning tasks. Note that the actual sign of a weight is not important, as the assignment of $+1$ and -1 to individual class labels is arbitrary and may change between individual learning tasks. We therefore use the absolute values $|w_j|$, or, more generally, the qth moment $|w_j|^q$ to assess feature relevance. More formally, we define:

$$\hat{\boldsymbol{\mu}} := \frac{1}{p}\sum_{i=1}^{p} |\boldsymbol{w}^i|^q, \qquad \boldsymbol{\mu} := \mathbb{E}\left[|\boldsymbol{w}^1|^q\right] = \ldots = \mathbb{E}\left[|\boldsymbol{w}^p|^q\right], \tag{3}$$

Here, $w^q := (w_1^q,\ldots,w_m^q)$ is meant to be the elementwise power. Note the $\boldsymbol{w}^i$ are i.i.d. and hence the definition of $\boldsymbol{\mu}$ can be made on base of any of the $\boldsymbol{w}^i$. As explained above, the qth moment $\hat{\boldsymbol{\mu}}$ measures how much information each component of the $\boldsymbol{w}^i$s has about the class label on average. For example, large components $\hat{\mu}_j$ correspond to large absolute values $|w_j^i|$, and hence the jth feature is likely to be discriminative—it is thus suggestive to employ a regularizer that promotes features with large μ_j. To promote features that are likely to be discriminative, we employ the following regularizer:

Moment-based Regularizer

$$B_b := B_b(T^1,\ldots,T^M) = \left\{\boldsymbol{w} \;\middle|\; \|\boldsymbol{w}\circ\hat{\boldsymbol{\mu}}^{-1}\| \le b\right\}, \quad \text{where } b > 0 \tag{B}$$

Here, $\circ$ denotes the elementwise multiplication of vectors, and we employ the notation $\boldsymbol{w}^{-1} = (1/w_1, \ldots, 1/w_m)$ to denote the elementwise inverse. Note that $\|\boldsymbol{w} \circ \hat{\boldsymbol{\mu}}^{-1}\|$ is only a shorter way to write $\sqrt{\sum_{j=1}^{m} w_j^2 / \hat{\mu}_j^2}$. Informally speaking, the regularizer (3) is an ℓ_2-norm regularizer, where the dimensions are scaled according to the moment of the corresponding features in the source data sets. Using this regularizer, we can state the proposed transfer approach as an easy three step procedure: First, obtain good weight vectors $\boldsymbol{w}^i$ on the source data sets, then compute the new regularizer B_b from the moments of the $\boldsymbol{w}^i$, and finally learn a target weight vector $\hat{\boldsymbol{w}}^\circ$ using B_b as regularizer. Note that the restriction to norm constraints is not a limitation since non-centered hypothesis classes are also subsumed by our analysis. This is because translations $\boldsymbol{w} \mapsto \boldsymbol{w}+\boldsymbol{t}$ cannot modify the Rademacher complexity by more than $\|\boldsymbol{t}\|_\infty/\sqrt{n}$ [3].

3 Theoretical Analysis

In this section we analyze the proposed transfer learning method theoretically in terms of upper bounds on the generalization error.

Theoretical performance measure. In order to theoretically measure the success of our approach, we compare its expected test error to one of the theoretically optimal linear classifier on the target data, i.e. we wish to obtain a bound of the form

$$\mathrm{er}^\circ(\hat{\boldsymbol{w}}^\circ) - \mathrm{er}^\circ(\boldsymbol{w}^*) \leq \text{bound},$$

$$\text{where } \ \boldsymbol{w}^* := \operatorname*{argmin}_{\boldsymbol{w}:\|\boldsymbol{w}\|\leq 1} \ \mathrm{er}^\circ(\boldsymbol{w}). \tag{4}$$

This bound compares the performance of our method to the one of the theoretically optimal vector $\boldsymbol{w}^*$. Of course, this quantity can not be observed, because the true underlying distribution is unknown. However, one can nevertheless obtain such an inequality by decomposing the above quantities into two terms as follows:

$$\begin{aligned} & \mathrm{er}^\circ(\hat{\boldsymbol{w}}^\circ) - \mathrm{er}^\circ(\boldsymbol{w}^*) \\ \leq\ & \mathrm{er}^\circ(\hat{\boldsymbol{w}}^\circ) - \hat{\mathrm{er}}^\circ(\hat{\boldsymbol{w}}^\circ) + \hat{\mathrm{er}}^\circ(\hat{\boldsymbol{w}}^\circ) \qquad (5) \\ & - \mathrm{er}^\circ(\boldsymbol{w}^\circ) + \mathrm{er}^\circ(\boldsymbol{w}^\circ) - \mathrm{er}^\circ(\boldsymbol{w}^*) \\ \leq\ & \underbrace{2 \sup_{\boldsymbol{w}\in B_b} \Big(|\mathrm{er}^\circ(\boldsymbol{w}) - \hat{\mathrm{er}}^\circ(\boldsymbol{w})|\Big)}_{\text{estimation error} \quad \mathrm{er}_e} + \underbrace{\mathrm{er}^\circ(\boldsymbol{w}^\circ) - \mathrm{er}^\circ(\boldsymbol{w}^*)}_{\text{approximation error} \quad \mathrm{er}_a}, \qquad (6) \end{aligned}$$

where we use the quantity $\boldsymbol{w}^\circ := \mathrm{argmin}_{\boldsymbol{w}\in B_b} \ \mathrm{er}^\circ(\boldsymbol{w})$ (this is the theoretical outcome of our approach (1) if we would optimize with regard to all n target examples instead of just the observed n° examples). To see that inequality (5) holds, note that $\hat{\mathrm{er}}^\circ(\hat{\boldsymbol{w}}^\circ) \leq \hat{\mathrm{er}}^\circ(\boldsymbol{w}^\circ)$. In the following, we address how to bound the estimation and approximation error separately.

3.1 Estimation Error

First, for the estimation error, we assume a fixed target regularizer B_b and only deal with the target data set. As explained in the error decomposition (6), it is sufficient to give a uniform convergence bound on the generalization error to bound the estimation error. To this aim, we give the following result:

Theorem 1. *Let the regularizer be as defined in* (B)*. Suppose the loss $\ell : \mathbb{R} \supset X \to [0,1]$ is Lipschitz with constant L, and the data lies in the unit cube, $\boldsymbol{x} \in [-1,1]^m$. Then, the following holds with probability larger than $1-\delta$:*

$$\sup_{\boldsymbol{w}\in B_b} \left|\mathrm{er}^\circ(\boldsymbol{w}) - \hat{\mathrm{er}}^\circ(\boldsymbol{w})\right| \leq \frac{2Lb}{\sqrt{n^\circ}} + 2\sqrt{\frac{2\ln\frac{2}{\delta}}{n^\circ}} \ .$$

The proof uses the established techniques and is shown in the Appendix. From (6) immediately follows for the estimation error $\mathrm{er}_e \leq 4L\frac{b}{\sqrt{n^\circ}} + 4\sqrt{\frac{2\ln\frac{2}{\delta}}{n^\circ}}$.

3.2 Approximation Error

The following theorems give upper bounds of the approximation error of the proposed approach (T) with B_b defined in (B). We start by giving a worst-case upper bound which does not make any assumption about the dataset-generating distribution $\mathcal{T}$.

Theorem 2. *Let the regularizer be defined as in* (B)*. Suppose the loss $\ell : \mathbb{R} \supset X \to [0,1]$ is Lipschitz with constant L, and the data lies in the unit cube, $\boldsymbol{x} \in [-1,1]^m$. Then, with probability greater than $1-\delta$ over the choice of the source data sets and with probability greater than $1-\epsilon$ over the choice of the target learning task,*[1] *the approximation error $\mathrm{er}_a = \mathrm{er}^\circ(\boldsymbol{w}^\circ) - \mathrm{er}^\circ(\boldsymbol{w}^*)$ has*

$$\mathrm{er}_a \leq \frac{Lm}{b^{\frac{1}{q}}}\left[\frac{1}{\epsilon}\left(\frac{q^q}{(q+1)^{q+1}} + b\sqrt{\frac{\ln\frac{m}{\delta}}{2p}}\right)\right]^{\frac{1}{q}} \tag{7}$$

Proof. We start the proof by noting that, since ℓ is Lipschitz with constant L, we know that $\ell(a) - \ell(b) \leq L|a-b|$ for all $a, b \in \mathbb{R}$. Thus we can use the Hölder inequality to bound the difference between the error of $\boldsymbol{w}^\circ$ and the error of $\boldsymbol{w}^*$ as follows:

$$\begin{aligned}
\mathrm{er}_a &= \mathrm{er}^\circ(\boldsymbol{w}^\circ) - \mathrm{er}^\circ(\boldsymbol{w}^*)\\
&= \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim P^\circ}\left[\ell(y{\boldsymbol{w}^\circ}^\top \boldsymbol{x})\right] - \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim P^\circ}\left[\ell(y{\boldsymbol{w}^*}^\top \boldsymbol{x})\right]\\
&\leq L \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim P^\circ}\left[\inf_{\boldsymbol{w}\in B_b}\left|\left(y\left(\boldsymbol{w}-\boldsymbol{w}^*\right)^\top \boldsymbol{x}\right)\right|\right]\\
&\leq L \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim P^\circ}\left[\|y\boldsymbol{x}\|_{\frac{q}{q-1}} \inf_{\boldsymbol{w}\in B_b}\|\boldsymbol{w}-\boldsymbol{w}^*\|_q\right]\\
&\leq Lm^{\frac{q-1}{q}} \inf_{\boldsymbol{w}\in B_b}\|\boldsymbol{w}-\boldsymbol{w}^*\|_q,
\end{aligned} \tag{8}$$

[1] In other words, with probability $(1-\epsilon)(1-\delta)$ over the combined distribution $\mathcal{T} \times P^\circ$.

where in the last step we exploit that the data lies in the unit cube, $\boldsymbol{x} \in [-1,1]^m$, and the labels are binary, $\boldsymbol{y} \in \{-1,1\}$. Note that $\inf_{\boldsymbol{w}\in B_b} \|\boldsymbol{w} - \boldsymbol{w}^*\|_q$ is a random variable because the optimal $\boldsymbol{w}^*$ depends on the draw of the target task T°. The above result (8) shows that in order to complete the proof, it suffices to show that with high probability (over the draw of the target task) $\inf_{\boldsymbol{w}\in B_b} \|\boldsymbol{w} - \boldsymbol{w}^*\|_q$ is smaller than the rightmost term in (7); i.e., we need: $\Pr\left[\inf_{\boldsymbol{w}\in B_b} \|\boldsymbol{w} - \boldsymbol{w}^*\|_q \geq t\right] \leq$ bound.

Before we proceed with this, we first need an auxiliary result: we show that the moment $\hat{\boldsymbol{\mu}}$ is concentrated around its expected value $\boldsymbol{\mu}$ (see definition in (3)). To this aim, consider the random variable $V_j := \mu_j - \hat{\mu}_j \overset{\text{Def.(3)}}{=} \mathbb{E}[w_j^{1\,q}] - \frac{1}{p}\sum_{i=1}^p w_j^{i\,q}$. Changing one source weight vector $\boldsymbol{w}^i$ changes the value of V_j by at most $\frac{1}{p}$. Thus, we can apply McDiarmid's inequality and obtain that $\Pr[V_j \geq t] \leq e^{-2t^2 p}$. Taking the union bound over all $V_j, 1 \leq j \leq m$ we get $\Pr[\max_j V_j \geq t] \leq m e^{-2t^2 p}$. Then, setting $t = \sqrt{\ln(m/\delta)/2p}$ yields

$$\Pr\left[\forall j:\ \ \mathbb{E}[|w_j^1|^q] - \frac{1}{p}\sum_{i=1}^p |w_j^i|^q \leq \sqrt{\frac{\ln\frac{m}{\delta}}{2p}}\right] \geq 1-\delta. \tag{9}$$

We are now ready to bound $\inf_{\boldsymbol{w}\in B_b} \|\boldsymbol{w} - \boldsymbol{w}^*\|_q$, which is the projection of $\boldsymbol{w}^*$ on B_b. Define $\boldsymbol{w}_B = \boldsymbol{w}^* \circ \min\left(\mathbf{1}, b\hat{\boldsymbol{\mu}}\right)$, where the min on a vector is understood elementwise. Using $\|\boldsymbol{w}^*\| \leq 1$, it is readily verified that $\boldsymbol{w}_B \in B_b$. Hence, defining $\boldsymbol{x}_+ := \max(\mathbf{0}, \boldsymbol{x})$, we have

$$\begin{aligned}
&\Pr\left[\inf_{\boldsymbol{w}\in B_b} \|\boldsymbol{w} - \boldsymbol{w}^*\|_q \geq t\right] \\
&\leq \Pr\left[\|\boldsymbol{w}_B - \boldsymbol{w}^*\|_q \geq t\right] \\
&= \Pr\left[\|\boldsymbol{w}^* - \boldsymbol{w}_B\|_q \geq t\right] \\
&= \Pr\left[\left\|\boldsymbol{w}^* \circ \left(\mathbf{1} - \frac{b}{p}\sum_{i=1}^p |\boldsymbol{w}^i|^q\right)_+\right\|_q \geq t\right] \\
&= \Pr\left[\sum_{j=1}^m \left|w_j^* \left(1 - \frac{b}{p}\sum_{i=1}^p |w_j^i|^q\right)_+\right|^q \geq t^q\right] \\
&\leq \frac{1}{t^q}\sum_{j=1}^m \underbrace{\mathbb{E}\left[|w_j^*|^q\right] \mathbb{E}\left[\left(1 - \frac{b}{p}\sum_{i=1}^p |w_j^i|^q\right)_+^q\right]}_{=:W_j}.
\end{aligned} \tag{10}$$

The last step is an application of Markov's inequality. We are now left with bounding the right hand side of (10).

We start by distinguishing between the cases $\mathbb{E}[|w_j^*|^q] \leq \sqrt{\ln\frac{m}{\delta}/\ 2p}$ and $\mathbb{E}[|w_j^*|^q] > \sqrt{\ln\frac{m}{\delta}\ /\ 2p}$. In the first case, we can use the bound $W_j \leq \mathbb{E}[|w_j^*|^q] \leq \sqrt{\ln\frac{m}{\delta}\ /\ 2p}$, because $\mathbb{E}\left[\left(1 - \frac{b}{p}\sum_{i=1}^p |w_j^i|^q\right)_+^q\right] \leq 1$. In the second case, we substitute (9) into the definition of W_j, so that it holds with probability larger than

$1-\delta$ that $W_j \le \mathbb{E}\left[|w_j^*|^q\right]\left(1 - b\,\mathbb{E}\left[|w_j^*|^q\right] + b\sqrt{\ln\frac{m}{\delta}/2p}\right)_+^q$. In both cases, W_j is no more than

$$W_j \le \mathbb{E}\left[|w_j^*|^q\right]\left(1 - b\left(\mathbb{E}\left[|w_j^*|^q\right] + \sqrt{\ln\frac{m}{\delta}/2p}\right)_+\right)_+^q.$$

We now proceed by bounding each W_j independently. Setting $a := \sqrt{\frac{\ln\frac{m}{\delta}}{2p}}$ and $x := \mathbb{E}\left[|w_j^*|^q\right]$, the above has the form $f(x) = x[1 - bx + ba]_+^q$, when restricting f to the interval $[a,1]$. A straightforward calculation shows that f has only one positive maximum at the position $x' = \frac{1+ab}{b+qb}$. If $ab > \frac{1}{q}$, then $x' < a$, so f reaches its maximum at the interval border $x = a$ with maximum value $f(x) = f(a) = a$. On the other hand, if $ab < \frac{1}{q}$, then $\boldsymbol{x}' > a$ and we can use $f(x') = \frac{q^q}{(q+1)^{q+1}}\frac{(1+ab)^{q+1}}{b}$ as an upper bound. In both cases f is not larger than $\frac{q^q}{b(q+1)^{q+1}} + a$. Re-substituting the definitions of a and x' we obtain $W_j \le \frac{q^q}{b(q+1)^{q+1}} + \sqrt{\frac{\ln\frac{m}{\delta}}{2p}}$. Plugging this into (10) yields

$$\begin{aligned}&\Pr\left[\inf_{\boldsymbol{w}_B \in B_b} \|\boldsymbol{w}^* - \boldsymbol{w}_B\| \ge t\right] \\ &\le \frac{m}{t^q}\left(\frac{q^q}{b(q+1)^{q+1}} + \sqrt{\frac{\ln\frac{m}{\delta}}{2p}}\right).\end{aligned} \tag{11}$$

The result follows by setting $t = \left[\frac{m}{\epsilon}\left(\frac{q^q}{b(q+1)^{q+1}} + \sqrt{\frac{\ln\frac{m}{\delta}}{2p}}\right)\right]^{\frac{1}{q}}$.

Of course, this inequality is loose in the sense that it gives non-trivial upper bounds only for cases where $b \ge m$. Ideally one would like to have $b < \sqrt{m}$, because this would improve the estimation error over the standard Rademacher bound, which is $O(\sqrt{m/n^\circ})$. The looseness is not surprising, because we have not made any assumption about the dataset-generating distribution $\mathcal{T}$. In the worst case, the distribution might have large variance, so that the source data sets do not contain any useful information about the target weight vector. However, it is easy to see how the result can be adapted to incorporate knowledge about $\mathcal{T}$. In the following, we give two results, where we make additional assumptions on $\mathcal{T}$.

3.3 Approximation Error with Sparse and Concentrated Moment Vectors

In the first case, we assume that only a fraction of the features are informative, so that some of the moment vector $\boldsymbol{\mu}$'s components can be bounded by a small constant. In the second case, we assume that the variance of the moment vectors is bounded.

Theorem 3. *Consider the same setting as in Theorem 2, but assume that there are $m_1 < m$ uninformative features, so that $\boldsymbol{\mu}_j \leq c$ for $1 \leq j \leq m_1$ and some small constant $c > 0$, while the remaining $m_2 := m - m_1$ features are informative, i.e. $\boldsymbol{\mu}_j$ is possibly larger than c for $m_1 < j \leq m$. Then, with probability greater than $1 - \delta$ over the choice of the source data sets and with probability greater than $1 - \epsilon$ over the choice of the target learning task, the approximation error can be upper-bounded by*

$$\text{er}_a \leq \frac{Lm^{\frac{q-1}{q}}}{b^{\frac{1}{q}}} \left[\frac{m_1 c}{\epsilon} + \frac{m_2}{\epsilon} \left(\frac{q^q}{(q+1)^{q+1}} + b\sqrt{\frac{\ln \frac{m}{\delta}}{2p}} \right) \right]^{\frac{1}{q}}$$

Proof. The proof follows the one of Theorem 2, but differs in the bound for the right hand side of (10). For the first $1 \leq j \leq m_1$ summands in (10), we can use $W_j \leq c$, because $\mathbb{E}\left[\left(1 - \frac{b}{p}\sum_{i=1}^{p} |w_j^i|^q \right)_+^q \right] \leq 1$. For the remaining m_2 summands, we use the original bound $W_j \leq \frac{q^q}{b(q+1)^{q+1}} + \sqrt{\frac{\ln \frac{m}{\delta}}{2p}}$. Plugging this into (10) yields

$$\Pr\left[\inf_{\boldsymbol{w}_B \in B_b} \|\boldsymbol{w}^* - \boldsymbol{w}_B\| \geq t \right] \leq \frac{m_1 c}{t^q} + \frac{m_2}{t^q} \left(\frac{q^q}{b(q+1)^{q+1}} + \sqrt{\frac{\ln \frac{m}{\delta}}{2p}} \right).$$

The result follows by setting $t = \left[\frac{m_1 c}{\epsilon} + \frac{m_2}{\epsilon} \left(\frac{q^q}{b(q+1)^{q+1}} + \sqrt{\frac{\ln \frac{m}{\delta}}{2p}} \right) \right]^{\frac{1}{q}}$.

The result leads to particularly tight bounds, if $q = 1$ and m_2 is small. It shows that one can achieve a comparably low generalization error, even if it is not known in advance, which features are informative and which ones are not. The following theorem deals with the case where all features are informative, but the moment vectors for all source and target data sets are concentrated sharply around the mean.

Theorem 4. *Consider the same setting as in Theorem 2, but assume that the variance of the moment vectors for the source and target data sets is bounded, that is, $\mathbb{E}[(\mu_j - \mathbb{E}[\mu_j])^2] \leq v$ is bounded by a constant v for all $1 \leq j \leq m$. Then, with probability greater than $1 - \delta$ over the choice of the source data sets and with probability greater than $1 - \epsilon$ over the choice of the target learning task, the approximation error can be upper-bounded by*

$$\text{er}_a \leq Lm \left[\frac{1}{\epsilon} \left(\frac{q^q}{b(q+1)^{q+1}} + \frac{\ln \frac{m}{\delta}}{3p} + \frac{\sqrt{\frac{4}{9}[\ln \frac{m}{\delta}]^2 + 8pv}}{2p} \right) \right]^{\frac{1}{q}} \tag{12}$$

If $v > 1$, the rightmost summand of the bound scales with $O(\sqrt{1/p})$, just as with the original result in Theorem 2. However, if v is close to zero, the two left summands are bounded by a $O(1/p)$ factor, leading to a significantly tighter bound.

Proof. The result follows by using the following inequality instead of (9) in the proof of Theorem 2:

$$\Pr\left[\forall j: \;\; \mathbb{E}[|w_j^1|^q] - \frac{1}{p}\sum_{i=1}^{p}|w_j^i|^q \leq \frac{\ln\frac{m}{\delta}}{3p} + \frac{\sqrt{\frac{1}{9}[\ln\frac{m}{\delta}]^2 + 2pv\ln\frac{m}{\delta}}}{p}\right] \geq 1-\delta. \tag{13}$$

To see that (13) holds, consider the random variable $V_j := \mu_j - \hat{\mu}_j \overset{\text{Def.(3)}}{=} \mathbb{E}[|w_j^1|^q] - \frac{1}{p}\sum_{i=1}^{p}|w_j^i|^q$. Since the variance of the moment vector's components is bounded by v, we can use Bernstein's inequality and obtain

$$\Pr[V_j \geq t] \leq \exp\left[\frac{pt^2}{2v + \frac{2t}{3}}\right].$$

Taking the union bound over all V_j, $1 \leq j \leq m$, we get

$$\Pr[\max_j V_j \geq t] \leq m\exp\left[\frac{pt^2}{2v + \frac{2t}{3}}\right].$$

Setting $t = \frac{1}{3p}\ln\frac{m}{\delta} + \frac{1}{p}\sqrt{\frac{1}{9}[\ln\frac{m}{\delta}]^2 + 2pv\ln\frac{m}{\delta}}$ yields (13).

Of course, there are many other possible assumptions about the learning task generating distribution $\mathcal{T}$ which would lead to non-trivial error bounds.

3.4 Discussion

We are now able to combine the bounds for the estimation and approximation error to obtain a bound for the total regret. For instance, setting $q = 1$ and using the setting in Theorem 3, we get the following Corollary:

Corollary 1. *Under the conditions of Theorem 3 it holds for the empirical risk minimizer $\boldsymbol{w}^\circ \in B_b$ defined in (1) and (8) for $q = 1$*

$$\mathrm{er}(\boldsymbol{w}^\circ) - \mathrm{er}(\boldsymbol{w}^*) \leq 4\sqrt{\frac{2\ln\frac{4}{\delta}}{n^\circ}} + L\left(\frac{4b}{\sqrt{n^\circ}} + \frac{m_1 c}{b\epsilon} + \frac{m_2}{b\epsilon}\left(\frac{1}{4} + \sqrt{\frac{\ln\frac{m}{\delta}}{2p}}\right)\right)$$

.

The bound can be expected to improve on standard regret bounds if c and the number of informative features m_2 are small, and the number of uninformative features m_1 is large. This is a very reasonable assumption as there are many problems in practice where only a few features are relevant and the large majority of features gets assigned only small weights (see, for example, [10]). In this scenario, we obtain a bound of the form $O(\frac{b}{\sqrt{n}} + \frac{m_1 c}{b} + \frac{m_2}{b\sqrt{p}})$ (omitting logarithmic factors), which can be considerably smaller than the $O(\sqrt{m/n})$ rate achieved by

Algorithm 1. *Moment-based transfer learning algorithm based on* (B) *and* (T)

```
 1: input target data set $T^\circ$ and source data sets $T^i = (X^i, Y^i)$, $i = 1, ..., p$
 2: for $i = 1$ to $p$
 3:    compute SVM weight vectors $\boldsymbol{w}^i := \mathrm{SVM}(X^i, Y^i)$ for source data sets $(X^i, Y^i)$,
    $i = 1, ..., p$,
           with SVM parameter $C$ tuned on a validation set
 4:    normalize each weight vector to unit norm: $\boldsymbol{w}^i := \boldsymbol{w}^i / \|\boldsymbol{w}^i\|_2$ for each $i = 1, \ldots, p$

 5: end for
 6: for various values of $q$
 7:    compute moment vector $\hat{\boldsymbol{\mu}} := \left(\frac{1}{p}\sum_{i=1}^p |\boldsymbol{w}^i|^q\right)$
 8:    reweight target training data: $\forall i = 1, \ldots, n:\ \boldsymbol{x}_i^\circ := \hat{\boldsymbol{\mu}}\boldsymbol{x}_i^\circ$
 9:    train SVM on target data set with parameter $C$ tuned on validation set
10:    denote the so-obtained weight vector by $\boldsymbol{w}_q^\circ$
11: end for
12: output the one SVM weight vector $\boldsymbol{w}_q^\circ$ with $q$ such that the error on the validation
    set is minimal
```

standard Rademacher-style concentration results. As a by-product, our analysis shows that the transfer learning approach is most beneficial when the sample size n is small and the dimensionality m is large. This is in accordance with anecdotal reports indicating that transfer learning is especially beneficial in small sample cases [15].

4 Algorithmic Details

In this section we describe the moment-based transfer learning algorithm based on (B) and (T) that we employed in the experiments in Section 5. To this aim, let us consider (B). It is easy to see that, instead of optimizing the original criterion (T), one can equivalently optimize a regular Support Vector Machine (SVM) [7] with the target data preprocessed by feature reweighting as follows: $\boldsymbol{x}_i^{\mathrm{new}} := \boldsymbol{x}_i^{\mathrm{old}} \circ \hat{\boldsymbol{\mu}}$. To see this, note that by employing a change of variables $\tilde{\boldsymbol{w}} = \boldsymbol{w} \circ \hat{\boldsymbol{\mu}}^{-1}$ it holds for the original criterion

$$\begin{aligned}\hat{\boldsymbol{w}}^\circ &\overset{(T)}{=} \underset{\boldsymbol{w}:\|\boldsymbol{w}\circ\hat{\boldsymbol{\mu}}^{-1}\|\le C}{\operatorname{argmin}} \frac{1}{n^\circ}\sum_{i=1}^{n^\circ} \ell(y_i^\circ \boldsymbol{w}^\top \boldsymbol{x}_i^\circ)\\ &= \underset{\tilde{\boldsymbol{w}}:\|\tilde{\boldsymbol{w}}\|\le C}{\operatorname{argmin}} \frac{1}{n^\circ}\sum_{i=1}^{n^\circ} \ell(y_i^\circ \tilde{\boldsymbol{w}}^\top (\boldsymbol{x}_i^\circ \circ \hat{\boldsymbol{\mu}}))\end{aligned}$$

The proposed method can now be stated as Algorithm 1. Lines 2–5 compute the optimal SVM weight vectors $\boldsymbol{w}^i$ on the source data sets and lines 6–11 perform the actual transfer learning on the target data set as defined on (B) and (T). In line 7 the transferred moments $\hat{\mu}$ are computed and in line 8–9 the actual transfer learning step is performed—as discussed above this is achieved by reweighting the features (line 8) and subsequently training an SVM on the so-obtained features (line 9). The final weight vector for the target task is output in line 12. The parameters q and C of our algorithm are tuned on a validation set.

5 Experiments

In this section we report on experiments with two application domains, text document classification and structure-activity-relationships. The first application domain, text document classification, is well suited for transfer learning because, even for very specialized topics, the Internet provides a large body of related source learning tasks. We downloaded ten data sets from TechTC, the Technion repository of text categorization data sets[2] [9]. Each data set contains between 142 and 277 text documents from two categories taken from the web directory *Open Directory Project*. In total this results in 1794 documents. The (binary classification) task is to tell for each data set the two categories apart. We employed a (binary) bag of words feature representation as provided by TechTC resulting in total in 142468 features.

To evaluate the predictive accuracy of the induced classifiers, we randomly split each data set into $r = 250$ training/validation/test partitions of size 50/25/25%. Then, we set one data set as target learning problem aside and kept the remaining data sets as source data. This process is repeated for each data set. Subsequently, we run Algorithm 1, a baseline (linear) SVM and the method of Evgeniou & Pontil [8] on the training partitions. For each repetition of the experiment, the optimal parameters were determined by a grid search over $q \in 10^{[-1,-0.8,\dots,1]}$ and $C \in 10^{[-3,2.5,\dots,4]}$ on base of the validation data set,[3] and test errors are computed on the test partition for the optimal parameter choices.

We give the results on the left hand side of Figure 1. The error bars indicate standard errors over the 250 repetitions. One can see that the method of Evgeniou & Pontil achieves an test error that is about 2% lower than the one of the SVM baseline for most data sets. The proposed method is on par with the SVM baseline for seven of the ten data sets and it is never worse than the SVM (as it contains the SVM as a special case for $q \approx 0$). For three data sets our method clearly outperforms the two other approaches with drastic gains in accuracy ranging from 13% to 19%.

In order to investigate why our method performed considerably better on these three particular data sets than on the remaining ones, we performed another experiment. We trained an SVM for each data set, using all feature vectors and all instances. This yields ten linear classifiers, that is, weight vectors $\hat{\boldsymbol{w}}^i$. We then compute the pairwise (absolute) correlation coefficients $\rho_{i,j} := |\mathrm{corr}(\hat{\boldsymbol{w}}^i, \hat{\boldsymbol{w}}^j)|$ for all $i, j = 1, \dots, 10$. The result is shown in Figure 1 (right). One can see that most tasks are only weakly correlated. This explains that our method did not improve

[2] The data sets are available at `http://techtc.cs.technion.ac.il/`.

[3] Optimal values of C were attained inside the grid. The second regularization parameter δ of the method of Evgeniou & Pontil was also determined by grid search: $\delta \in 10^{[-2,-1,\dots,2]}$.

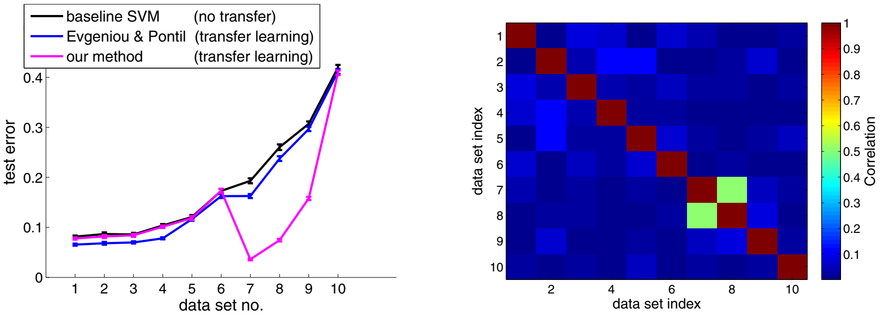


Fig. 1. Empirical results of the text categorization experiment: test errors (left) and correlation coefficients of the SVM weights (right). Vertical bars indicate standard errors. One can see that the correlation is maximized for the data set pair (7,8)—this accordance with the test errors: the gain in accuracy of our method over the baselines is maximal on these data.

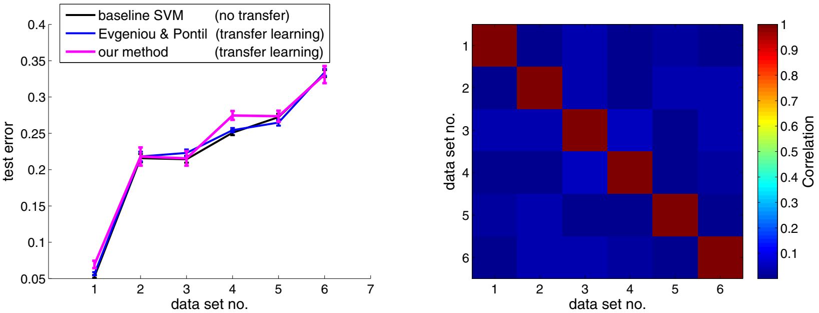


Fig. 2. Empirical results of the structure-activity relationship experiment

over the baselines on most data sets—one might conjecture that the corresponding tasks are only weakly related. However, the correlation is substantially stronger for the data set pair (7,8): it is $\rho_{7,8} = 0.51$ while the second largest coefficient only has $\rho_{2,9} = 0.11$. This observation is in accordance with our empirical results: the gain in accuracy over the baselines was the highest for data sets 7 and 8 (19% and 16%, respectively)—thus, we conclude that those two tasks are very closely related and this is why our method works best in these cases.

We also performed some experiments with learning the biological activity of compounds given their molecular structure as a graph. We obtained the six datasets used in [14]. Each dataset contains a number of molecular graphs and about 1000 features testing for the presence of one frequently occurring substructure. We again randomly split each data set into $r = 100$ training/validation/test partitions of size 50/25/25%, evaluated the proposed approach and compared it to the baseline approaches as done above. It turns out that there is no gain in using the proposed method for these data sets (see Figure 2). A closer investigation indicates that there are too many substructure features, which are distinct between the individual tasks to make feature-level transfer suitable. This is true both for our approach as the one by Evgeniou & Pontil. It is an interesting open question whether one could use feature description data to transfer information between distinct, but similar features.

6 Discussion and Conclusion

In this paper we presented a transfer learning approach for adjusting the regularizer of a target learning problem. This is an important task for many of the modern machine learning applications, where the features often outnumber the training instances. Empirical results have shown that it is often not enough to impose strong standard regularizers (e.g. to encourage sparsity), but that individual learning problems benefit from customized regularization [2,8,13,6,16]. The results in this paper demonstrate that adaptive regularization can be successfully applied to transfer information from source to target data sets. The main idea is to extend an ℓ_2-norm regularizer with feature weights and to transfer good values for these weights from the source data sets. A theoretical analysis showed that the expected prediction error depends critically on the trade-off between estimation and approximation error. If the source classifiers are close to the optimal target classifier, then it is possible to keep the approximation error small simply by choosing a strong regularizer that penalizes weight vectors too far away from the source classifiers. The empirical analysis on real text classification data shows that our approach works well in practice if the dataset share transferable information: for some data sets a gain in accuracy of up to 19% was observed while it never performed worse than the SVM baseline. It is an open question whether the bounds can be improved for special cases, and if other parametrization approaches lead to better theoretical or practical results.

Acknowledgments. UR and MK acknowledge a grant by the German National Science Foundation (DFG Ru 1589/1-1) and a scholarship by the German Academic Exchange Service (DAAD), respectively. This work was supported in part by the European Community under the PASCAL2 Network of Excellence (ICT-216886).

References

1. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. Journal of Machine Learning Research 6, 1817–1853 (2005)
2. Argyriou, A., Evgeniou, T., Pontil, M.: Convex multi-task feature learning. Machine Learning 73(3), 243–272 (2008)
3. Bartlett, P.L., Mendelson, S.: Rademacher and gaussian complexities: Risk bounds and structural results. JMLR 3, 463–482 (2002)
4. Baxter, J.: A model of inductive bias learning. Journal of Artificial Intelligence Research 12, 149–198 (2000)

5. Ben-David, S., Schuller, R.: Exploiting task relatedness for mulitple task learning. In: Proceedings of the 16th Annual Conference on Computational Learning Theory, pp. 567–580 (2003)
6. Caruana, R.: Multitask learning. Mach. Learn. 28, 41–75 (1997)
7. Cortes, C., Vapnik, V.N.: Support vector networks. Machine Learning 20, 273–297 (1995)
8. 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, New York (2004)
9. Gabrilovich, E., Markovitch, S.: Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In: Proceedings of The 27th Annual International ACM SIGIR Conference, Sheffield, UK, pp. 250–257. ACM Press, New York (2004)
10. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182 (2003)
11. Maurer, A.: Bounds for linear multi-task learning. J. Mach. Learn. Res. 7, 117–139 (2006)
12. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 99 (2009) (PrePrints)
13. Raina, R., Ng, A.Y., Koller, D.: Constructing informative priors using transfer learning. In: ICML 2006: Proceedings of the 23rd International Conference on Machine Learning, pp. 713–720. ACM, New York (2006)
14. Rückert, U., Kramer, S.: Kernel-based inductive transfer. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 220–233. Springer, Heidelberg (2008)
15. Schweikert, G., Widmer, C., Schölkopf, B., Rätsch, G.: An empirical analysis of domain adaptation algorithms for genomic sequence analysis. In: Advances in Neural Information Processing Systems, vol. 21, pp. 1433–1440 (2009)
16. Zhong, E., Fan, W., Peng, J., Verscheure, O., Ren, J.: Universal learning over related distributions and adaptive graph transduction. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009. LNCS, vol. 5782, pp. 678–693. Springer, Heidelberg (2009)
17. Zhong, E., Fan, W., Peng, J., Zhang, K., Ren, J., Turaga, D.S., Verscheure, O.: Cross domain distribution adaptation via kernel mapping. In: Knowledge Discovery and Data Mining, pp. 1027–1036 (2009)

A Proof of Theorem 1

Let $S_n = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$ be a set of independent Rademacher variables, which obtain the values -1 or +1 with the same probability 0.5. Then, the *Rademacher complexity* of the class of linear classifiers with regularizer B_b is given by $\mathcal{R}_B := \mathbb{E}_{S_n}\left[\left|\sup_{\boldsymbol{w}\in B_b} \frac{1}{n}\sum_{i=1}^{n} s_i \boldsymbol{w}^\top \boldsymbol{x}_i^\circ\right|\right]$. We will now give an upper bound for the Rademacher complexity of the moment-based approach to transfer learning.

Proposition 1. *Then, the Rademacher complexity of linear classifiers with B_b regularization as defined in* (3) *is upper-bounded by:*

$$\mathcal{R}_B \leq \frac{b}{\sqrt{n}} \; .$$

Proof. By employing variable substitutions of the form $\boldsymbol{v} = b^{-1}\boldsymbol{w} \circ \hat{\boldsymbol{\mu}}^{-1}$ we can use the Cauchy-Schwarz (C-S) inequality to bound the Rademacher complexity $\mathcal{R}_B$ as follows:

$$\begin{aligned}
\mathcal{R}_B &= \mathbb{E}_{S_n}\left[\sup_{\boldsymbol{v}:\|\boldsymbol{v}\|_2\le 1}\left|\frac{1}{n}\sum_{i=1}^{n} s_i\left(b\boldsymbol{v}\circ\hat{\boldsymbol{\mu}}\right)^\top \boldsymbol{x}_i^\circ\right|\right] \\
&\overset{\text{C}-\text{S}}{\le} \mathop{\mathbb{E}}_{S_n}\left[\sup_{\boldsymbol{v}:\|\boldsymbol{v}\|_2\le 1}\|\boldsymbol{v}\|\left\|\frac{b}{n}\sum_{i=1}^{n} s_i\left(\hat{\boldsymbol{\mu}}\circ\boldsymbol{x}_i^\circ\right)\right\|\right] \\
&= \frac{b}{n}\mathbb{E}_{S_n}\left[\sqrt{\sum_{i,j=1}^{n} s_i s_j\left(\hat{\boldsymbol{\mu}}\circ\boldsymbol{x}_i\right)^\top\left(\hat{\boldsymbol{\mu}}^q\circ\boldsymbol{x}_j\right)}\right] \\
&\le \frac{b}{n}\sqrt{\sum_{i,j=1}^{n}\mathbb{E}_{S_n}\left[s_i s_j\left(\hat{\boldsymbol{\mu}}\circ\boldsymbol{x}_i\right)^\top\left(\hat{\boldsymbol{\mu}}\circ\boldsymbol{x}_j\right)\right]} \\
&= \frac{b}{n}\sqrt{\mathbb{E}_{S_n}\sum_{i=1}^{n}\|\hat{\boldsymbol{\mu}}\circ\boldsymbol{x}_i\|^2} \;\le\; \frac{b}{n}\sqrt{\mathbb{E}_{S_n}\sum_{i=1}^{n}\|\hat{\boldsymbol{\mu}}\|^2}
\end{aligned}$$

where for the third step we use that the Rademacher variables are independent, and in the forth step that the data is in $[-1,1]^m$. Recall that $\|\boldsymbol{w}^i\|_2 \le 1$ for all i and thus $\||\boldsymbol{w}^i|^q\|_2 \le 1$. Hence,

$$\|\hat{\boldsymbol{\mu}}\| \le \|\frac{1}{p}\sum_{i=1}^{p}|\boldsymbol{w}^i|^q\| \le \frac{1}{p}\sum_{i=1}^{p}\||\boldsymbol{w}^i|^q\| \le 1.$$

Combining this with the above bound gives the claimed result.

If the Rademacher complexity of a class of classifiers is known, it can be used to bound the generalization error:

Theorem 5 ([3]). *Suppose the loss $\ell : \mathbb{R} \supset X \to [0,1]$ is Lipschitz with constant L. Then, the following holds with probability larger than $1-\delta$:*

$$\sup_{\boldsymbol{w}\in B_b}\left|\mathrm{er}^\circ(\boldsymbol{w}) - \hat{\mathrm{er}}^\circ(\boldsymbol{w})\right| \;\le\; 2L\mathcal{R}_B + 4\sqrt{\frac{2\ln\frac{4}{\delta}}{n}}\;.$$

Theorem 1 follows now from combining the previous two results.

Multimodal Nonlinear Filtering Using Gauss-Hermite Quadrature

Hannes P. Saal, Nicolas M.O. Heess, and Sethu Vijayakumar

School of Informatics, University of Edinburgh, Edinburgh, Scotland, UK
{hannes.saal,sethu.vijayakumar}@ed.ac.uk, n.m.o.heess@sms.ed.ac.uk

Abstract. In many filtering problems the exact posterior state distribution is not tractable and is therefore approximated using simpler parametric forms, such as single Gaussian distributions. In nonlinear filtering problems the posterior state distribution can, however, take complex shapes and even become multimodal so that single Gaussians are no longer sufficient. A standard solution to this problem is to use a bank of independent filters that individually represent the posterior with a single Gaussian and jointly form a mixture of Gaussians representation. Unfortunately, since the filters are optimized separately and interactions between the components consequently not taken into account, the resulting representation is typically poor. As an alternative we therefore propose to directly optimize the full approximating mixture distribution by minimizing the KL divergence to the true state posterior. For this purpose we describe a deterministic sampling approach that allows us to perform the intractable minimization approximately and at reasonable computational cost. We find that the proposed method models multimodal posterior distributions noticeably better than banks of independent filters even when the latter are allowed many more mixture components. We demonstrate the importance of accurately representing the posterior with a tractable number of components in an active learning scenario where we report faster convergence, both in terms of number of observations processed and in terms of computation time, and more reliable convergence on up to ten-dimensional problems.

1 Introduction

Filtering is a common problem in robotics and other areas where observations become available sequentially. The observations have a stochastic dependence on an unobserved underlying state and the goal is to infer the posterior distribution over the state at a particular time step given the observations up to that time step. In settings where the observation function is nonlinear the posterior state distribution can take on complex shapes and even become multimodal. For example, in visual tracking, observations are often ambiguous due to other moving objects or occlusion. In such cases, the posterior distribution might comprise a relatively small number of reasonably well isolated modes, each of which could be modeled well by a single Gaussian distribution. Representing such multimodal distributions as a whole with a single Gaussian distribution, however, can

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 81–96, 2011.

lead to divergence of the filter estimates and generally poor performance. Properly representing posterior state distributions, including their multimodality, is especially important when using active learning methods since the uncertainty captured by the posterior is used directly in the decision of how to query the next observation in order to resolve ambiguities in the state as quickly as possible.

Although filtering approaches that rely on mixtures of Gaussians to represent a skewed or multimodal state distribution have a long history, most of these approaches rely on banks of linear filters, each with a Gaussian state distribution, that are updated independently [2]. While this has the advantage of being computationally very efficient, since interactions between the mixture components are being ignored, the resulting mixture distribution is likely to be a poor fit of the true underlying state distribution. This can lead to poor overall performance unless a large number of mixture components is used.

Yet, in many situations a small number of Gaussian components can be sufficient to capture the essential structure of the posterior distribution if the parameters of the mixture components are chosen appropriately. This leads us to explore new ways of optimizing the parameters of the approximate mixture representation of the posterior distribution: We present a novel approach to the problem of mixture filtering that takes inspiration from variational approaches to approximate inference and combine this with a deterministic sampling approach: We assume that the prior (e.g. the filtering distribution from the previous time step) is given as a mixture of Gaussians (MoG). Due to this MoG prior, and due to a nonlinear observation function the posterior distribution over the state given a new observation can have a complex shape. We therefore approximate the new posterior distribution again as a MoG distribution. We optimize this approximate posterior distribution by approximately minimizing the Kullback-Leibler (KL) divergence between the true updated state distribution and the approximate MoG representation. Exact minimization of the KL divergence is intractable. Our approximate minimization relies on a deterministic sampling approach, Gauss-Hermite Quadrature, which evaluates general integrals by evaluating the integrand at suitably chosen sample points, and we describe a novel way to re-formulate the required integrals so as to optimize the accuracy of the method for the problem at hand.

2 Methods

2.1 Problem Statement

We are interested in filtering problems, but in this paper ignore the time update and instead focus on the measurement update.[1] At each time step we receive a new observation $\mathbf{y}_t$ and use this to update our current estimate of the latent state $\mathbf{x}_t$: $p(\mathbf{x}_t|\mathbf{y}_t, \mathbf{y}_{1\ldots t-1}) \propto \int \mathrm{d}\mathbf{x}_{t-1} p(\mathbf{y}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1\ldots t-1})$ where

[1] This corresponds to a static target. Including a dynamics model is straightforward and time updates could be done as in other mixture filters by propagating each mixture component independently.

$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \delta(\mathbf{x}_t - \mathbf{x}_{t-1})$. For any time step this problem can be thought of as computing the posterior $p(\mathbf{x}|\mathbf{y})$ using the state distribution from the *previous* time-step as the prior $p(\mathbf{x})$. As discussed above, for most interesting cases the true posterior cannot be computed exactly. This is the case, when the likelihood $p(\mathbf{y}|\mathbf{x})$ is not of the convenient linear-Gaussian form. The focus of this paper is on developing a formulation of the filtering problem that allows for an approximate representation of the state distribution given previous observations that is sufficiently flexible to account for uncertainty in the latent state e.g. when the true posterior is multimodal or skewed. Since in our scenario the posterior computed in step t will be the prior for step $t+1$ we are interested in a representation of the posterior that can be directly used as the prior in the calculations for the next time step. Specifically, we will be representing the prior and the approximate posterior as a MoG distribution. Furthermore, we assume that the observation likelihood is a Gaussian with fixed covariance, but with a mean that depends on $\mathbf{x}$ via a nonlinear function $f(\cdot)$ which we choose to represent as a radial basis function (RBF) network. We choose this form because it allows us to treat terms arising from the likelihood analytically (cf. Sec. 2.2) and at the same time is general enough to approximate any nonlinear function to arbitrary accuracy [15]. Alternative formulations, e.g. using a Gaussian process representation are also conceivable [5][2]. Thus, at each time step we are faced with the following problem: Given

$$p(\mathbf{x}) = \sum_n \gamma_n p_n(\mathbf{x}) \tag{1}$$

$$p_n(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\nu}_n, \boldsymbol{\Theta}_n) \tag{2}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{f}(\mathbf{x}), \boldsymbol{\Sigma}_y) \tag{3}$$

$$f(\mathbf{x}) = \sum_j c_j k(\mathbf{x}, \mathbf{m}_j) \tag{4}$$

$$k(\mathbf{x}, \mathbf{m}_j) = \exp\left\{-0.5(\mathbf{x} - \mathbf{m}_j)^T \mathbf{S}^{-1} (\mathbf{x} - \mathbf{m}_j)\right\} \tag{5}$$

our goal is to approximate the state posterior $p(\mathbf{x}|\mathbf{y})$ with a mixture of Gaussians

$$q_{\text{mix}}(\mathbf{x}) = \sum_m \alpha_m q_m(\mathbf{x}) \tag{6}$$

$$q_m(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m). \tag{7}$$

2.2 Fitting a Gaussian Mixture to the Posterior

Given a new observation $\mathbf{y}$, and the current prior $p(\mathbf{x})$ we need to optimize the parameters of $q(\mathbf{x})$ so as to match $p(\mathbf{x}|\mathbf{y})$ as closely as possible. The variational

[2] If the observation function is given in analytical form, expected values can instead be estimated by the using linearization methods from commonly used unimodal filters, like the extended Kalman filter or the unscented filter.

approach requires the Kullback-Leibler (KL) divergence between the approximate posterior $q(\mathbf{x})$ and the true posterior to be minimized

$$\mathrm{KL}\left[q||p\right] = \int \mathrm{d}\mathbf{x}\ q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x}|\mathbf{y})}. \tag{8}$$

The choice of $\mathrm{KL}\left[q||p\right]$ can be motivated by the fact that the resulting approximate posterior leads to a lower bound on the log marginal likelihood $\log p(X)$ of a latent variable model $p(X, Z)$:

$$\log p(X) = \log \int \mathrm{d}Z\ p(X,Z) \geq \int \mathrm{d}Z\ q(Z) \log \frac{p(X,Z)}{q(Z)} \tag{9}$$

where the difference between the left-hand and the right-hand side is just the KL divergence between the approximate posterior q and the true posterior $p(Z|X)$. Minimizing this divergence tightens the bound (it becomes tight iff q is equal to the true posterior, in which case the KL divergence is zero). Maximizing this bound with respect to the model parameters allows for maximum likelihood learning in models with intractable posteriors.

In order to be able to capture complex shapes of the true posterior, including multimodality, we choose MoG as our approximate posterior distribution. In our case $\mathrm{KL}\left[q_{\mathrm{mix}}||p\right]$ can be broken down as follows:

$$\begin{aligned}\mathrm{KL}\left[q_{\mathrm{mix}}||p\right] &= \int \mathrm{d}\mathbf{x}\ q_{\mathrm{mix}}(\mathbf{x}) \log \frac{q_{\mathrm{mix}}(\mathbf{x})}{p(\mathbf{x}|\mathbf{y})} \qquad (10)\\ &= -\,\mathrm{H}\left[q_{\mathrm{mix}}\right] - \sum_m \alpha_m \mathrm{E}_{q_m}\left[\log p(\mathbf{x})\right]\\ &\quad - \sum_m \alpha_m \mathrm{E}_{q_m}\left[\log p(\mathbf{y}|\mathbf{x})\right] + \mathrm{const} \qquad (11)\end{aligned}$$

where $\mathrm{H}\left[q\right] = -\int \mathrm{d}\mathbf{x}\ q(\mathbf{x}) \log q(\mathbf{x})$ is the differential entropy of q and the expectations $\mathrm{E}_{q_m}\left[\cdot\right]$ are taken with respect to the Gaussian components q_m of the mixture posterior.

In order to obtain the approximate posterior this expression needs to be minimized with respect to the parameters of q_{mix}. With the choices made above for prior, likelihood, and approximate posterior (equations 1–6) we can exactly compute the third term in (11) (see Appendix A.1) but the first and second term are not tractable since they involve integrals taken over log-sums. We approximate these intractable integrals in (11) using quadrature methods as described in the next section.

2.3 Gauss-Hermite Quadrature

Gauss-Hermite quadrature [1] approximates d-dimensional integrals by deterministically selecting sample points from a weight function—in this case a Gaussian $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$—and then computing a weighted sum of the function values at those sample points:

$$\int \mathrm{d}\mathbf{x}\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)f(\mathbf{x}) = \mathrm{E}_{q_m}\left[f(\mathbf{x})\right] \approx \pi^{-\frac{d}{2}} \sum_{\mathbf{h}} w_{\mathbf{h}} f(\mathbf{z_h}) \tag{12}$$

where $w_{\mathbf{h}} = \prod_d w_{\mathbf{h}(d)}$, i.e. the overall weights are determined as the product of the individual single dimension weights. $\mathbf{z_h}$ are the transformed sample points $\mathbf{z_h} = \mathbf{L}_m\mathbf{x_h}\sqrt{2} + \boldsymbol{\mu}_m$, where $\mathbf{L}_m\mathbf{L}_m^T = \boldsymbol{\Sigma}_m$. In this paper, we set $\mathbf{L}_m$ to be the Cholesky factor, but any triangular decomposition of $\boldsymbol{\Sigma}_m$ could be used (cf. [3]). The sample points and corresponding weights are given by the roots of the Hermite polynomial and can be calculated offline and stored. Derivatives of the resulting approximation are straightforward to calculate. In our setup, the function $f(\cdot)$ is given as an RBF and thus the integral has the following form:

$$\mathrm{E}_{q_m}\left[f\right] = \mathrm{E}_{q_m}\left[\log \sum_n \alpha_n \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)\right] \tag{13}$$

Gauss-Hermite quadrature could be used to approximate this integral directly (see [7] for such an approach), but we found that the estimate can become highly inaccurate if the variances of the individual mixture components differ considerably and only a small number of sample points is used. This can then lead to a divergence of the optimization of the KL-divergence. Instead of increasing the number of sample points, which quickly becomes untenable in high dimensions, we instead rewrite the integral as a sum, which allows us to approximate each term of this sum individually. This should allow for higher accuracy, as we can optimize the sample points for each term separately. Thus, we write $\mathrm{E}_{q_m}\left[f\right]$ as:

$$\mathrm{E}_{q_m}\left[f\right] = \mathrm{E}_{q_m}\left[\log \alpha_1\mathcal{N}_1\right] + \sum_{n=2}^{N} \mathrm{E}_{q_m}\left[\log\left(1 + \frac{\alpha_n\mathcal{N}_n}{\sum_{k=1}^{n-1}\alpha_k\mathcal{N}_k}\right)\right] \tag{14}$$

where we use the abbreviation $\mathcal{N}_n$ to stand in for the longer $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$. The first component of this sum can be calculated analytically, while the remaining ones have to be approximated. We use different weighting functions for each of these terms. For each integral, we approximate it in the standard way as described above, if the variance of $\mathcal{N}_m$ is smaller than the variance of $\mathcal{N}_n$. Otherwise, we rewrite the integral and choose $\mathcal{N}_n$ instead of $\mathcal{N}_m$ as the weighting function:

$$\underbrace{\mathrm{E}_{q_m}\left[\log\left(1 + \frac{\alpha_n\mathcal{N}_n}{\sum_{k=1}^{n-1}\alpha_k\mathcal{N}_k}\right)\right]}_{\text{Integral over } q_m} = \underbrace{\mathrm{E}_{q_n}\left[\frac{\mathcal{N}_m}{\mathcal{N}_n}\log\left(1 + \frac{\alpha_n\mathcal{N}_n}{\sum_{k=1}^{n-1}\alpha_k\mathcal{N}_k}\right)\right]}_{\text{Integral over } q_n} \tag{15}$$

As illustrated in Fig. 1 this makes it more likely that the sample points will capture the region that is interesting for integration, as the mean and variance of the new weighting function should be closer to the mode and log curvature of the integrand, and thereby improving accuracy [14]. In the multivariate case, we either pick the component with the lowest covariance determinant, or (when restricting ourselves to diagonal covariance matrices) treat each dimension independently.

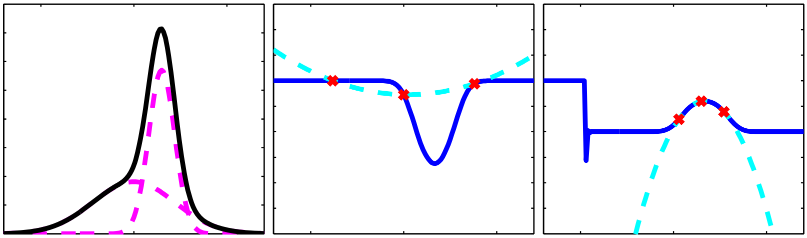


Fig. 1. Illustration of the improved Gauss-Hermite method. Left: Mixture of two Gaussians, q_1 (left) and q_2 (right). Middle: Sample points from q_1 (red), integrand (dark blue) and implicit polynomial fit to integrand by quadrature method (light blue) when integrating according to the left-hand term in Eq. (15). Right: Improved fit in relevant region (around sample points) when integrating according to the right-hand term in Eq. (15), with sample points taken from q_2.

3 Related Work

As explained in the introduction, Gaussian mixture distributions as an approximation to the true state distribution have a long history in the filtering literature. The classic approach to Gaussian mixture filtering uses a weighted sum of extended Kalman filters (EKFs) running in parallel [2]. Newer approaches replace the EKFs with linear filters using deterministic sampling [10,3]. However, in all these cases several unimodal filters run independently of each other in parallel. While this is computationally very efficient, it also leads to inferior representation of the posterior distribution. Different ways to compute the posterior mixture weights have also been proposed, in an attempt to decrease the distance between the true posterior and the mixture approximation [10]. In contrast, in our approach we adapt *all* Gaussian mixture parameters (i.e. means, covariances, and mixture weights) *jointly* so as to fit the true posterior as closely as possible.

Mixture distributions as approximate posterior distributions have been considered previously in the literature on variational inference [11]. In particular Lawrence and Azzouzi[13], as well as Bouchard and Zoeter [4] consider the use of MoGs to approximate continuous-valued posterior distributions. To our knowledge, these have, however, not been considered in the context of filtering. Compared to previous work, the filtering application leads to an additional intractable term in (11) in the form of the integral over the logarithm of the prior which, in our case, does not have a simple parametric form such as Gaussian, but rather is a MoG itself. Previous work deals with the intractable terms differently: Jaakola and Jordan [11], who consider the case of discrete distributions, employ an advanced upper bound to approximate the expectation of the logarithm arising in the expectation of $\mathrm{H}\left[q_{\mathrm{mix}}\right]$. This bound requires the optimization of additional variational parameters including a set of “smoothing distributions”. Lawrence and Azzouzi [13] adapt this approach to the continuous case using MoGs for the posterior. The variational smoothing distributions then take the form of Gaussians whose parameters need to be optimized alongside the other variational parameters. This happens in an iterative scheme which alternates gradient ascent with respect to the different sets of parameters.

In this paper we are primarily interested in a fast method for estimating the approximate representation of the posterior. Instead of minimizing an upper bound on the KL divergence as in [11,13] we therefore approximate the intractable integrals in (11) using quadrature methods which (a) avoid the need to iteratively optimize additional parameters and are therefore fast, and (b) directly extend to the second intractable term to which the upper bound to the logarithm used for evaluating the entropy is not applicable.

Gauss-Hermite quadrature has been used in unimodal Gaussian filtering before to approximate certain integrals [10,3]. The well-known unscented filter [12] also relies on deterministic sampling, but uses a slightly different approach to selecting the sample points and weights. In all these cases, deterministic sampling is used in order to calculate expectations over the observation function. In our approach, we use deterministic sampling in order to approximate expectations over log-sums, i.e. entropy-like terms and their gradients. While entropies of mixtures of Gaussians have been approximated using deterministic sampling before [7], our way differs in that we treat each component *inside* the log-sum independently, and therefore achieve higher accuracy. Another way that has been proposed would be to derive the Taylor expansion of the log-sum up to a certain degree and then integrate analytically [9].

4 Results

4.1 Problem Setup

In order to test the performance of our proposed variational approach, we examined how well it could fit complex (i.e. multimodal or skewed) state distributions when compared with other approaches. To highlight the importance of such representations, we additionally tested whether an improved posterior representation would help in a localization task with ambiguous observations, while using active learning in order to speed up convergence.

Depending on task difficulty, we compared our approach with a number of other options: first, a linear mixture filter[3] using the same number of components as the variational mixture filter; second, linear filters using a higher number of components (3^d or 7^d)[4]; and finally, a variational filter using just a single component, in order to examine how well a unimodal approach would perform[5].

[3] Our linear mixture filter implementation is a bank of independent filters that are updated independently. Note that because of the particular form of the nonlinear likelihood (RBF, cf. Eqs. (4), (5)) all required expectations can be computed analytically (see Appendix A.1).

[4] d denotes the dimensionality of the state distribution.

[5] The unimodal variational filter still needs to be optimized iteratively, but since the posterior distribution only consists of a single Gaussian, the KL divergence and its gradient can be evaluated analytically, so no numerical quadrature is needed and optimization is usually much quicker.

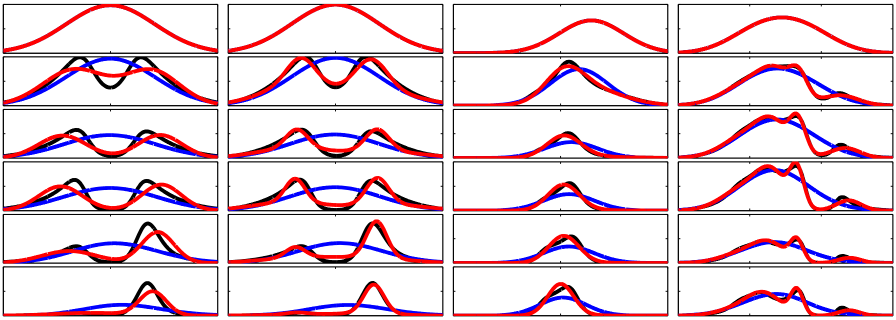


Fig. 2. Comparison of true posterior (black lines), variational (red) and 'bank of filters' (blue) approximations for different observation functions. Each row represents a new observation, with successive time steps ordered from top to bottom, while each of the four columns stands for a different problem: 1. Bimodal posterior, approximated with two components. 2. Same posterior as in (1), approximated with 4 components, 3. Skewed posterior, 2 components, 4. Highly multimodal posterior, 4 components.

The problem was set up as a localization task where the position of a (stationary) target at location $\mathbf{x}^*$ had to be estimated by probing search locations $\boldsymbol{\theta}$ sequentially and receiving observations $\mathbf{y}$ that depended on both the probe and target locations: $\mathbf{y} = f(\boldsymbol{\theta} - \mathbf{x}^*)$. Each run started with a wide Gaussian prior (zero-mean with an isotropic variance of 9), reflecting the fact that the location of the target was unknown. The observation function $f(\cdot)$ was set up as an RBF consisting of a small number of individual squared exponential components. In each new run, the location of these components with respect to the target location was sampled from a uniform distribution over a hyper-cube of length 8 centred at the origin. The number and kernel width varied with the dimensionality of the problem: We used 3 kernels with a width of 1.5 for the 4D problem, 4 kernels with width 1.5 for 6D, 5 kernels with width 3 for 8D, and 5 kernels with width 7 for 10D. We used two different types of observations: Ambiguous and Infomax observations. For ambiguous ones, search locations $\boldsymbol{\theta}$ were fixed such that observations always came from the mode of an individual RBF component, which frequently resulted in the posterior becoming multimodal. This type of observations was used to test how well the different methods were able to model multimodal state distributions (see results in Sec. 4.2). For Infomax observations, the search locations $\boldsymbol{\theta}$ were optimized via active learning such that they resulted in the biggest information gain about the position of the target (see Appendix A.2 for mathematical details and Sec. 4.3 below for results). These observations should allow the posterior to quickly converge onto the correct target. That is, knowing the observation function as well as the current (possibly multimodal) state distribution allows selecting search locations that disambiguate between the different potential target positions effectively.

For the cases where the linear filter used a higher number of components than the variational one, its prior was initialized to match the original prior as closely

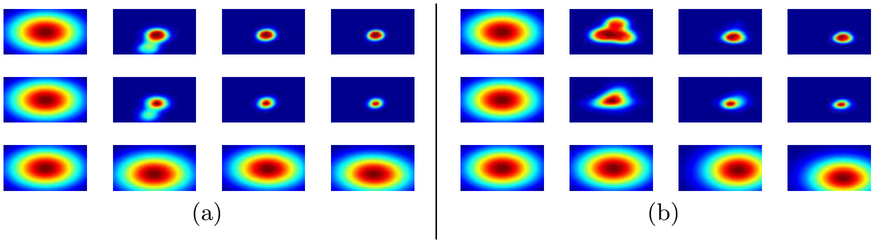


Fig. 3. (a) and (b). Two example runs with time increasing from left to right (4 steps each). Top row: Actual posterior (calculated numerically). Middle row: Variational mixture approximation (3 components). Bottom row: Linear mixture approximations (3 components). All methods start with the same prior and receive the same observations.

as possible by placing components on a grid and adjusting their weights so that the resulting mixture distribution matched the original broad Gaussian prior distribution. We also tried other initializations, e.g. randomly sampling components from the original prior, and found that different initializations did not influence the results much. For the variational approach, we used 3 quadrature points per dimension in all examples, leading to 3^d samples in total. For problems where $d > 2$, we restricted the variational method to diagonal covariance matrices; components in linear filters always maintained full covariance matrices, however. All algorithms were implemented in Matlab, using some functions from the Lightspeed toolbox[6]. For gradient descent we used the scaled conjugate gradient implementation provided by the Netlab toolbox[7].

4.2 Representation of Multimodal Posterior Distributions

First, we tested how well our approach could represent skewed and multimodal posterior state distributions of different dimensionalities. Some examples for different setups in 1D and 2D are shown in Figs. 2 and 3. As can be seen, the variational approximation generally approximates the posterior well and correctly finds and represents the major modes. The quality of approximation evidently improves with the number of mixture components that is used (see second column in Fig. 2). Moreover, skewed distributions can be fitted well, by using several mixture components (see e.g. the third column in Fig. 2). Sometimes the variational approximation covers several posterior modes with a single component, which also happens when there are more posterior modes than mixture components (see right panel in Fig. 3). The 'bank of filters' method on the other hand runs into problems if a unimodal prior splits into a posterior consisting of several components, and often retains an excessively high variance.

[6] http://research.microsoft.com/en-us/um/people/minka/software/lightspeed/

[7] http://www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/

We also ran a more exhaustive test on low-dimensional problems (1–2D), where we kept the observation function constant, but systematically varied the mean and the (co)variance of the prior distribution with respect to the observation function. This introduced a big range of different nonlinearities for the methods to encounter. We ran the algorithms on several different observation functions by varying the number and locations of RBF components. For each individual trial, we (numerically) calculated the KL divergence between the distributions approximated by the mixture methods and the true state posterior. We found that, generally, our algorithm was at least as good as the linear mixture filter but often dramatically better, usually when the posterior state distribution became either considerably skewed or multimodal.

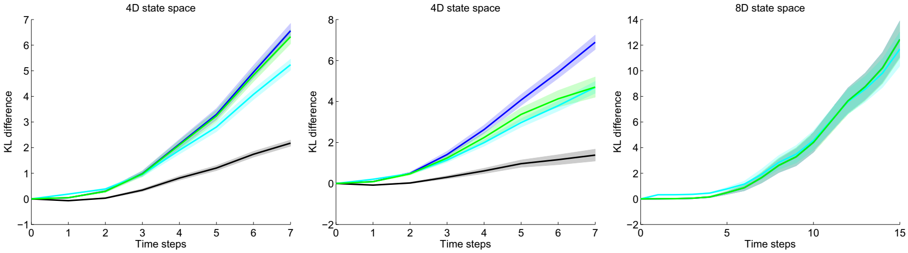


Fig. 4. Differences between the KL divergence between respective posterior approximation and the true posterior, and the KL divergence between the variational filter and the true posterior as a function of the number of observations. Positive values indicate that the variational mixture filter was closer to the true posterior (in terms of the KL divergence), while negative values denote the respective other filter being a better fit. Left: 4D problem with observation function consisting of a single bump. Middle: 4D problem with three observation function modes. Right: 8D problem with 5 modes. Blue line: Average difference between linear and variational mixture filters (both using 4 mixture components). Green line: Difference between unimodal (1 component) and variational filter). Light blue and black lines (4D only): Same for linear filters using 81 and 2401 components, respectively. Shaded regions indicate standard error of the mean.

In a further set of tests, we examined how well the different mixture methods were able to capture high-dimensional complex posterior distributions. In order to quantify the fit of the different representations, we calculated differences in the KL divergences between the different mixture approximations and the true posterior distributions over time: we used Monte-Carlo integration in order to arrive at an estimate of the KL divergence (up to a constant). In these tests we only presented ambiguous samples as we were interested in a complex posterior shape. Results for tasks in 4D (with either a single or three observation function modes) and 8D (using 5 modes) can be seen in Fig. 4. We ran this task for 100 (4D) or 25 (8D) random configurations of the observation function and target. We noticed several interesting effects. First, a linear filter using the same number of components as the variational filter consistently performs worse, independent of whether the posterior state distribution is skewed (left panel) or becomes multimodal (middle panel). Second, adding more components to the linear filter improves the difference in KL divergence. However, even with a very large number of components (2401), performance is generally much worse than the variational mixture filter. Also, the number of components that would be needed quickly becomes infeasible in higher dimensions ($d > 4$). For example, using 7 mixture components per dimension would have required more than 5 million components in 8 dimensions, which exceeds the memory limitations of our setup. Finally, a unimodal variational filter cannot represent the complex posterior shapes properly. We noticed that most of the time, the unimodal version tends to cover all of the posterior modes, although in some cases it could "fall" into a single posterior mode, leaving other ones uncovered.

4.3 Convergence When Using Active Learning

In another set of simulations, we used active learning in order to quickly resolve the uncertainty introduced by ambiguous observations. For this, we optimized successive search locations with respect to their informativeness about the target location (see Appendix A.2 for mathematical details). This was to highlight the importance of multimodal representations in a practical scenario. In these tests we first presented a number of ambiguous samples (3–5), causing the posterior distribution to acquire a complex shape and possibly become multimodal. We then iterated between optimizing the next search location and updating the state distribution after receiving an observation from that location. This optimization crucially depended on the prior state distribution at the current time step. Using active learning should quickly resolve any ambiguities in the state distributions and lead to quick convergence of the posterior distribution to the actual target. Methods better at representing multimodal posteriors should converge more quickly as they should be better at correctly representing the uncertainty in the state space. In this part of the analysis, we did not examine linear filters with a bigger number of components than the variational filter, as the runtime of our active learning framework is quadratic in the number of Gaussian mixture components (see Appendix A.2), and therefore prohibitively slow to use with a lot of individual components.

We examined how well the different algorithms converged onto the target location for both 6D and 10D problems. Fig. 5 shows both the mean squared error (MSE) as well as the root covariance determinant over time[8]. Additionally we plot the log likelihood of the actual target over time. Convergence is indicated by both decreasing error and root covariance determinant, while the log likelihood of the target should increase over time. We found that the variational mixture filter converged well towards the actual target over time, while both the linear and unimodal filters seemed to stall. This means that the active learning component could exploit the multimodal representation of the variational approach

[8] The MSE and mean root covariance determinant were calculated with respect to the overall mean and covariance of the approximate mixture distribution.

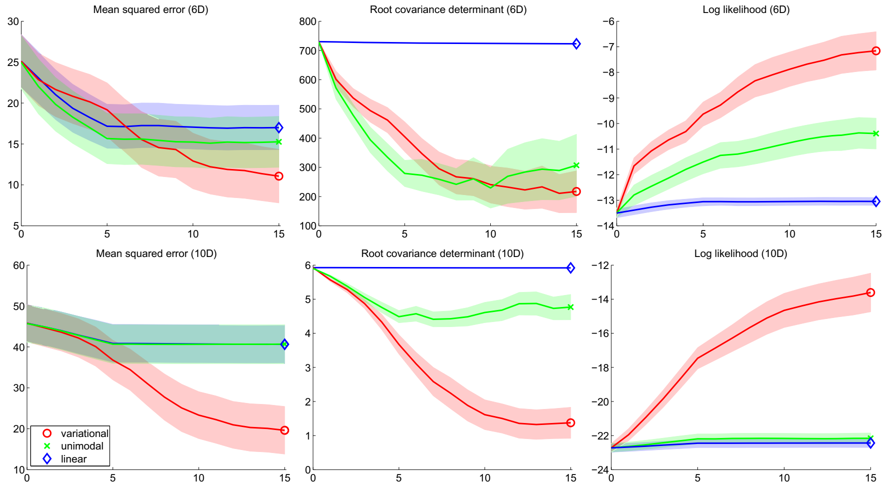


Fig. 5. Convergence results for different algorithms on a 6D (top row of panels) and 10D (bottom row) problem, respectively, with the linear and variational algorithm using 3 components each. Left panels: Average mean squared error over 15 or 20 time steps, respectively, for 25 runs of the variational (red), linear (blue), and unimodal variational (green) algorithms. Shaded regions indicate standard error of the mean. Both multimodal approaches used 4 mixture components. Middle panels: Average covariance determinant over time. Right panels: Log likelihood of the actual target over time.

and resolve the ambiguity about the target location. The representation of the state distribution by the other methods, however, was not sufficient to allow for effective target localization.

As the variational mixture filter is computationally more involved, it is slower in updating the posterior state distribution after receiving an observation due to the numerical optimization involved. We asked whether these delays would have any effects on performance. We therefore set a fixed time span (30–90 seconds), during which each of the methods would iteratively determine the next sample point using active learning, then receive an observation and update its posterior state distribution. The time needed for both optimization of the next search location (i.e. the active learning part) and updating the posterior distributions counted towards each method's time budget. Thus, the faster an algorithm updated its state representation and the lower its number of mixture components, the more observations it was able to request. Fig. 6 plots the number of observations that was used by each method against the log likelihood at the end of the time span for a 3D (left) and a 6D problem (right). In this plot, a marker in the left, upper corner indicates that the respective method was only able to request a small number of observations, but achieving a high log likelihood. Markers in the right, lower corner, however, would indicate that an algorithm requested a high number of observations but failed to increase the log likelihood considerably. We found that the variational mixture filter does

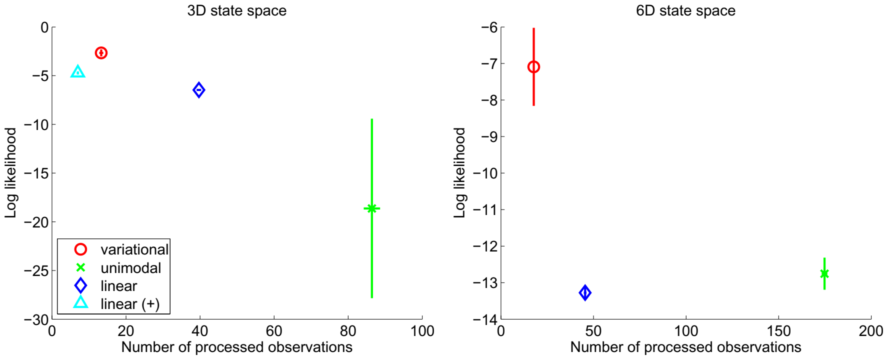


Fig. 6. Log likelihood of target after updates plotted against number of processed observations. Markers denote mean (over 25 runs each) and vertical lines indicate standard errors of the mean. Left: 3D state space. Right: 6D state space. A fixed time budget of 30 and 90 seconds, respectively, was imposed per algorithm. Light blue marker (left plot only): Linear filter with 27 components. Red: Variational mixture filter (3 components). Blue: Linear filter (3 components). Green: Unimodal variational filter.

well in both tasks, by increasing the log likelihood more than other methods, despite being relatively slow and therefore only processing a small number of observations. The linear filter with 27 components also does well in the 3D example, but performs even slower due to increased computational demands in the active learning stage. The variational unimodal filter is generally the fastest, but does not perform well. The extremely high standard error observed is due to its mode-seeking behavior, which caused it to model only a single posterior mode, which often turned out to be the “wrong” one.

5 Discussion

In this paper we have proposed a novel approach to filtering in which the approximate posterior distribution over the state is maintained as a mixture of Gaussians. Using a MoG to approximately represent the posterior makes it possible to capture complex shapes of the true state distribution such as skewedness or multimodality which often arise when the observation function is nonlinear. Unlike previous approaches to mixture filtering we do not maintain a set of independent Gaussian components but take interactions between the mixture components into account when optimizing the approximate representation of the posterior distribution given a new observation. This requires the calculation of expectations over log-sums, which cannot be done analytically, and we propose to approximate these terms using quadrature methods. We find that optimizing the mixture representation directly captures the true shape of the posterior much better than a bank of independent linear filters, even when allowing many more components. We demonstrate the impact of this improved representation in a task where active learning is used in order to directly resolve the uncertainty in the posterior distribution resulting from ambiguous samples: The proposed approach converges considerably faster and more reliably than the alternative filtering approaches. Importantly, faster convergence is achieved not just when measured as a function of observations but also in terms of overall compute time, despite the fact that our filtering approach is computationally more expensive than the alternatives: The additional computational complexity in processing individual observations by the filter is more than compensated for by the noticeably faster convergence per observation as demonstrated in the experiments with limited overall run-time.

In this paper we have focused on a mixture of Gaussians representation in order to capture multimodal or skewed posterior distributions. Sampling methods like the particle filter have been proposed as an alternative to traditional filtering methods when dealing with nonlinear observation functions and the resulting multimodal state distributions. They work by maintaining a large collection of weighted samples (particles) that provide a sample based representation of the posterior distribution. While such a sample based representation can in principle approximate distributions of any shape (including multi-modal distributions), the number of particles required increases exponentially so that they become impractical in high-dimensional spaces. Furthermore, for many applications a compact representation of the posterior e.g. in terms of a small number of mixture components is crucial: For instance, in the context of the application discussed in this paper, active learning, the sample-based methods that have been proposed so far are very slow even with a relatively small number of particles, which renders them feasible in very low-dimensional spaces only [8,16].

There are several interesting directions for future work. Firstly, our method uses an observation function that is represented as an RBF network. A relatively straightforward extension would be to use a Gaussian Process representation instead [5]. In cases where the observation function is given directly in analytic form, it might not be possible to calculate expectations over this function. In such cases, expectations can be estimated using methods from unimodal Gaussian filters, such as in the extended Kalman filter or in the unscented filter [12]. Secondly, an important practical consideration is the number of mixture components used to represent the posterior. In the experiments described above this number was fixed. Using too many components does not negatively impact the quality of the posterior representation, but it does slow down the algorithm. Using as few components as possible is therefore desirable in order to achieve fast convergence in terms of compute time. On the other hand, choosing the number of components too small will leave some part of the state space unrepresented or requires a single component to cover multiple posterior modes, which will result in a worse representation and in the extreme case can lead to similar problems as for a unimodal filtering approach. One interesting approach would be to dynamically adjust the number of components, adding new components in each observation step, and then merging the most similar ones.

References

1. Abramowitz, M., Stegun, I.A.: Handbook of mathematical function with formulas, graphs, and mathematical tables. U.S. Dept. of Commerce (1972)
2. Alspach, D., Sorenson, H.: Nonlinear bayesian estimation using Gaussian sum approximations. IEEE Trans. Autom. Control 17(4), 439–448 (1972)
3. Arasaratnam, I., Haykin, S., Elliott, R.J.: Discrete-time nonlinear filtering algorithms using Gauss-Hermite quadrature. Proc. IEEE 95(5), 953–977 (2007)
4. Bouchard, G., Zoeter, O.: Split variational inference. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, pp. 57–64. ACM, New York (2009)
5. Deisenroth, M.P., Huber, M.F., Hanebeck, U.D.: Analytic moment-based Gaussian process filtering. In: ICML (2009)
6. Girard, A., Rasmussen, C.E., Candela, J.Q., Murray-Smith, R.: Gaussian process priors with uncertain inputs—applications to multiple-step ahead time series forecasting. In: NIPS (2003)
7. Goldberger, J., Gordon, S., Greenspan, H.: An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures. In: Proc. IEEE Int. Conf. on Computer Vision (ICCV), vol. 1, pp. 487–493 (2003)
8. Hoffmann, G., Waslander, S., Tomlin, C.: Mutual information methods with particle filters for mobile sensor network control. In: Proc. IEEE Conf. on Decision and Control, pp. 1019–1024 (2006)
9. Huber, M.F., Bailey, T., Durrant-Whyte, H., Hanebeck, U.D.: On entropy approximation for Gaussian mixture random vectors. In: Proc. IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems, pp. 181–188 (2008)
10. Ito, K., Xiong, K.: Gaussian filters for nonlinear filtering problems. IEEE Trans. Autom. Control. 45(5), 910–927 (2000)
11. Jaakola, T.S., Jordan, M.I.: Improving the mean field approximation via the use of mixture distributions. In: Jordan, M.I. (ed.) Learning in Graphical Models, pp. 163–173. Kluwer Academic Publishers, Dordrecht (1998)
12. Julier, S., Uhlmann, J.: Unscented filtering and nonlinear estimation. Proc. IEEE 92(3), 401–422 (2004)
13. Lawrence, N.D., Azzouzi, M.: A variational Bayesian committee of neural networks. Technical report, University of Cambridge, UK (1999)
14. Liu, Q., Pierce, D.A.: A note on Gauss-Hermite quadrature. Biometrika 81(3), 624–629 (1994)
15. Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. Neural Comput. 3, 246–257 (1991)
16. Saal, H.P., Ting, J., Vijayakumar, S.: Active sequential learning with tactile feedback. In: AISTATS (2010)
17. Torkkola, K.: Feature extraction by non-parametric mutual information maximization. J. Mach. Learn. Res. 3, 1415–1438 (2003)

A Appendix

A.1 Additional Calculations

Usually, expectations over the observation function f have to be approximated, however if f is represented as a RBF as in our case, or as a Gaussian Process [5], then these expectations can be calculated analytically (e.g. [6]), as given below:

$$\mathrm{E}_{q_{mix}}\left[f(\mathbf{x})\right] = \sum_m \sum_j \mathrm{E}_{q_m}\left[c_j k(\mathbf{x}, \mathbf{m}_j)\right] \tag{16}$$

$$\mathrm{E}_{q_m}\left[c_j k(\mathbf{x}, \mathbf{m}_j)\right] = c_j |\mathbf{S}^{-1}\boldsymbol{\Sigma}_m + \mathbf{I}|^{-\frac{1}{2}} \cdot \tag{17}$$

$$\exp\left\{-0.5(\boldsymbol{\mu}_m - \mathbf{m}_j)^T(\boldsymbol{\Sigma}_m + \mathbf{S})^{-1}(\boldsymbol{\mu}_m - \mathbf{m}_j)\right\} \tag{18}$$

$$\mathrm{E}_{q_{mix}}\left[f(\mathbf{x})^2\right] = \sum_m \sum_i \sum_j \mathrm{E}_{q_m}\left[c_i k(\mathbf{x}, \mathbf{m}_i) c_j k(\mathbf{x}, \mathbf{m}_j)\right] \tag{19}$$

$$\mathrm{E}_{q_m}\left[c_i k(\mathbf{x}, \mathbf{m}_i) c_j k(\mathbf{x}, \mathbf{m}_j)\right] = c_i c_j |2\mathbf{S}^{-1}\boldsymbol{\Sigma}_m + \mathbf{I}|^{-\frac{1}{2}} \tag{20}$$

$$\exp\left\{-0.5(\boldsymbol{\mu}_m - \hat{\mathbf{m}}_{ij})^T(\boldsymbol{\Sigma}_m + 0.5\mathbf{S})^{-1}(\boldsymbol{\mu}_m - \hat{\mathbf{m}}_{ij})\right\} \tag{21}$$

A.2 Active Learning

In the active learning scenario, our aim is to pick a search location $\boldsymbol{\theta}$, which maximizes the mutual information between the current state distribution and the expected observation. As the mutual information cannot be calculated analytically when distributions are represented as mixtures of Gaussians, we instead optimize a surrogate measure, called 'quadratic mutual information' (QMI) that has been originally proposed for clustering [17].

$$I_{QMI}(X;Y|\Theta) = \iint \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}\big(p(\mathbf{x},\mathbf{y}|\boldsymbol{\theta}) - p(\mathbf{x})p(\mathbf{y}|\boldsymbol{\theta})\big)^2 \tag{22}$$

$$= \iint \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}\ p(\mathbf{x},\mathbf{y}|\boldsymbol{\theta})^2 + \iint \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}\ p(\mathbf{x})^2 p(\mathbf{y}|\boldsymbol{\theta})^2 - \tag{23}$$

$$2\iint \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}\ p(\mathbf{x},\mathbf{y}|\boldsymbol{\theta})p(\mathbf{x})p(\mathbf{y}|\boldsymbol{\theta}) \tag{24}$$

Each of the integrals involved can now be calculated analytically in a similar fashion as described in Appendix A.1. At each step, I is optimized by gradient ascent with respect to the new search location $\boldsymbol{\theta}$. The computational complexity of this approach is quadratic in the number of mixture components.

Active Supervised Domain Adaptation

Avishek Saha[1,⋆], Piyush Rai[1⋆], Hal Daumé III[2],
Suresh Venkatasubramanian[1], and Scott L. DuVall[3]

[1] School of Computing, University of Utah
{avishek,piyush,suresh}@cs.utah.edu
[2] Department of Computer Science, University of Maryland CP
hal@umiacs.umd.edu
[3] VA SLC Healthcare System & University of Utah
scott.duvall@hsc.utah.edu

Abstract. In this paper, we harness the synergy between two important learning paradigms, namely, active learning and domain adaptation. We show how active learning in a target domain can leverage information from a different but related source domain. Our proposed framework, Active Learning Domain Adapted (ALDA), uses source domain knowledge to transfer information that facilitates active learning in the target domain. We propose two variants of ALDA: a batch B-ALDA and an online O-ALDA. Empirical comparisons with numerous baselines on real-world datasets establish the efficacy of the proposed methods.

Keywords: active learning, domain adaptation, batch, online.

1 Introduction

We consider the *supervised*[1] domain adaptation setting [9] where we have a large amount of labeled data from some source domain, a large amount of unlabeled data from a target domain, and *additionally* a small budget for acquiring labels in the target domain. We show how, apart from leveraging information in the usual domain adaptation sense, the information from the source domain is *further* leveraged to selectively query for labels in the target domain (instead of choosing them randomly, as is the common practice). We achieve this by first training the best possible classifier in the source without using target labels, for instance, either by simply training a supervised classifier on the source labeled data, or by using some unsupervised adaptation technique using the unlabeled target data as well. Then, we use this learned hypothesis in various ways to leverage the source domain information when we are additionally given some fixed budget for acquiring some extra *labeled* target data (i.e., the active learning setting [12]).

[⋆] Authors contributed equally.

[1] We define *supervised domain adaptation* as having labeled data in both *source* and *target*, *unsupervised domain adaptation* as having labeled data in only *source*, and *semi-supervised domain adaptation* as having labeled data in *source* and both labeled and unlabeled data in *target*.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 97–112, 2011.

We call this framework Active Learning Domain Adapted (Alda). Our proposed framework is based on three key components. The first component is *unsupervised* domain adaptation (i.e., without target labeled data). The goal of this step is to suitably adapt the source data representation such that it makes the marginal distributions of both source and target distributions look similar. This enables training any traditional supervised classifier for the target domain using the adapted representation of the source data. The second and the third components improve this classifier even further by using active learning to selectively acquire the labels of target examples, given a budget on the target labels. Moreover, these components leverage the source domain information as well. Specifically, the second step employs a *domain separator hypothesis* that rules out querying labels of those target examples that appear "similar" to examples from the source domain. The domain separator hypothesis is a classifier that distinguishes between source and target domain examples and *is learned using only unlabeled examples from the two domains.* The third component is a hybrid oracle which consists of two oracles: one that provides labels for free but is imperfect (there could be noise), and one expensive (but "perfect") oracle used in the standard active learning settings. The source classifier acts as the free oracle which, although not perfect, can provide correct labels for most of the examples queried (essentially, the ones that appear 'source' like).

The proposed Alda framework is sufficiently general to allow varied choices of domain adaptation and active learning modules. In addition, Alda applies to both batch (Section 2) as well as online settings (Section 3). In this paper, we present empirical results (Section 4) for specific choices of the domain adaptation and the active learning schemes. For both batch and online settings, we empirically demonstrate that the proposed approach leads to significant improvement in prediction accuracies for a given target label budget, when compared to other baselines. Moreover, for the online setting, apart from showing empirically better performance, we also show that our approach results in smaller mistake bounds under suitable notions of domain separation. We provide intuitive arguments for smaller label complexity in the target domain when compared to the standard active learning where we do not have access to data from a related distribution.

2 Alda: Active Learning Domain Adapted

In this section, we propose a principled approach towards active learning in a target domain by leveraging information from a related source domain. In our setting, we are given a small budget for acquiring labels in a target domain, which makes it imperative to use active learning in the target domain. However, our goal is to *additionally* leverage the domain relatedness by exploiting whatever information we might already have from the source domain. At a high level, our proposed approach aims to answer the following questions:

1. given source information, which samples in the *target* are the most informative (in an *active sense*)?
2. among the *informative target samples*, can we use source information to *infer labels* of a few *informative target samples*, such that the actual number of target labels queried (from an oracle) is reduced even further?

In the following, we provide answers to the above questions. We begin by introducing some notations and presenting an overview of the ALDA framework.

2.1 Preliminaries

Let $\mathcal{X} \subset \mathbb{R}^d$ denote the instance space and $\mathcal{Y} = \{-1, +1\}$ denote the label space. Let $\mathcal{D}_s(x, y)$ and $\mathcal{D}_t(x, y)$ be the joint source and target distributions, respectively. We have a set of source labeled examples $L_s(\sim \mathcal{D}_s(x, y))$ and a set of source unlabeled examples $U_s(\sim \mathcal{D}_s(x))$. Additionally, we also have a set of target unlabeled instances $U_t(\sim \mathcal{D}_t(x))$, from which we *actively* acquire labels. Furthermore, $\mathbf{w}_{src}$ denotes a classifier learned from the source labeled data and $\mathbf{w}_{ds}$ denotes the *domain separator* hypothesis. Finally, let ϕ represent an unsupervised domain adaptation algorithm that outputs a classifier $\mathbf{u}_\phi$. Note that learning $\mathbf{u}_\phi$ *does not require* labeled target examples.

Fig. 1 shows our basic setup for ALDA. The Active Learning (AL) module is a combination of the sub-modules Uncertainty Sampler (US) (that is initialized using the $\mathbf{u}_\phi$ classifier from the unsupervised domain adaptation phase) and Domain Separator (DS) (that uses the $\mathbf{w}_{ds}$ classifier). In addition, the setup employs

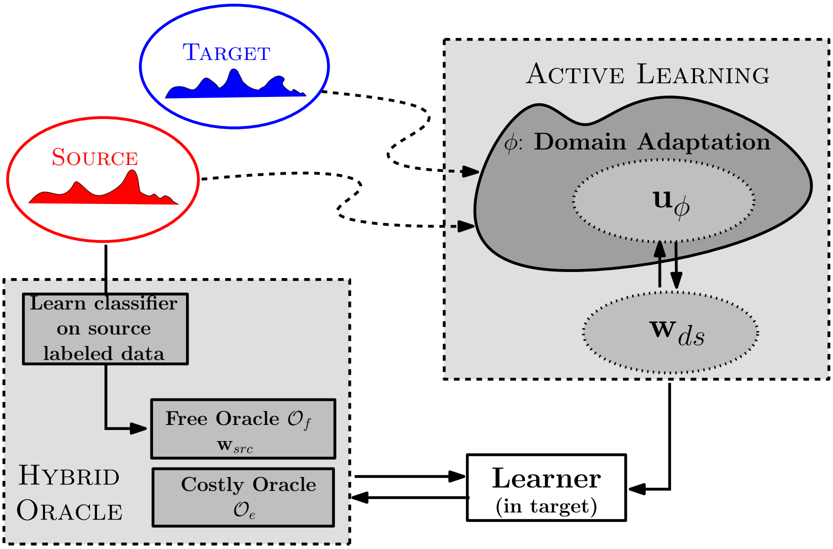


Fig. 1. An illustration of the proposed ALDA framework. Domain adaptation can be performed using any black-box unsupervised domain adaptation approach (e.g., [2,14]). The active learning block can be any batch or online active learner.

a *hybrid oracle* which is a combination of a free oracle $\mathcal{O}_f$ and an expensive oracle $\mathcal{O}_c$. The free oracle $\mathcal{O}_f$ is nothing but the classifier ($\mathbf{w}_{src}$) learned using the source labeled samples L_s. At each step, the learner *actively* selects an informative target sample and gets it labeled by an *appropriate* oracle. This continues in an iterative (for the batch setting) or online fashion until some stopping criterion is met (say, for example, reached prescribed accuracy or exhausted label budget). Next we describe each of these individual modules in more detail.

2.2 Initializing the Uncertainty Sampler

The first phase of ALDA learns an *unsupervised* domain adapted classifier $\mathbf{u}_\phi$ which uses labeled source data, and unlabeled source and target data. Note that this phase does not use any labeled target data (hence the name unsupervised). There are a number of ways to learn the classifier $\mathbf{u}_\phi$. In this paper, we take the approach [14] that is based on estimating the *importance ratio* between the source and the target distribution, without actually estimating these distributions. The source domain examples, with their corresponding importance weights, can then be used to train any classifier which is now readily adapted for the target domain (of course, this can potentially still be improved, given extra labeled target data). Note that the unsupervised domain adaptation step can be performed using a number of other ways as well; for example, Kernel Mean Matching (KMM) can be performed by matching the source and target distributions in some Reproducing Kernel Hilbert Space (RKHS) and computing the importance weights of source domain examples [8]. Another approach (especially for NLP problems), could be to use Structural Correspondence Learning (SCL) to identify invariant ("pivot") features between source and target, and use these features for unsupervised domain adaptation [2]. The unsupervised domain adapted classifier $\mathbf{u}_\phi$ serves as the initializing classifier for the subsequent active learning phase of our approach.

2.3 Leveraging Domain Divergence

It turns out that, in addition to using the source domain information to initialize our active learner in the target domain (Section 2.2), we can further leverage the domain relatedness information to improve the active learning phase in the target. In this section, we propose the *domain separator* that further leverages the relatedness of source and target domains while performing active learning in the target. Assuming the source and target domains to be related, our proposed technique exploits this relatedness to upfront rule out acquiring labels of those *target domain examples* that "appear" to be close to the source domain.

As an example, Fig. 2 shows a typical domain separator hypothesis (denoted by $\mathbf{w}_{ds}$) that separates the *source* and *target* examples. We note that similar source and target examples are expected to have the same labels since only the marginal distribution of examples changes between the source and target examples (i.e., $\mathcal{D}_s(x) \neq \mathcal{D}_t(x)$) whereas the conditional distribution of labels (given the examples) stays the same (i.e., $\mathcal{D}_s(y|x) = \mathcal{D}_t(y|x)$). Observe that if the source and target distributions are far apart, then the two domains can be

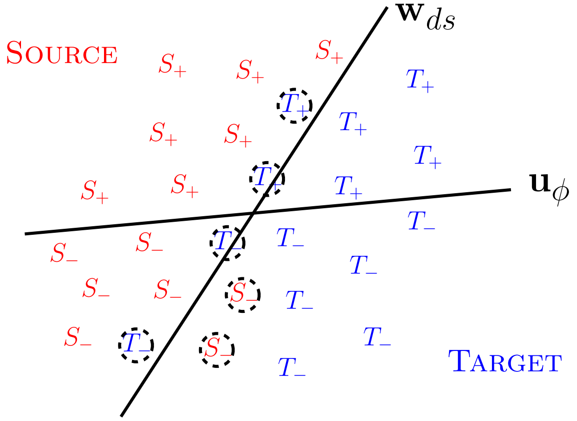


Fig. 2. An illustrative diagram showing the domain separator hypothesis $\mathbf{w}_{ds}$ separating source data from target data and the classifier $\mathbf{u}_\phi$ learned using the unsupervised domain adapted source classifier.

perfectly classified by this separator. However, if the domains are similar, it is expected that there will be a reasonable overlap and therefore some of the target (or source) domain examples might lie on the source (or target) side (encircled instances in Fig. 2) and hence will be misclassified by the domain separator hypothesis. Acquiring labels for such target domain examples (that lie on the source side) is not really needed since the initial hypothesis (refer $\mathbf{u}_\phi$ in Fig. 1) of ALDA would already have taken into account such examples. Therefore, such target examples can be effectively ignored from being queried. Thus the domain separator hypothesis, which can be learned using *only* source and target *unlabeled* examples, provides a novel way of performing active sampling in domain adaptation settings.

The domain separator hypothesis avoids querying the labels of all those target examples that lie on the source side of the domain separator and hence are misclassified by it. This number, in turn, depends on the domain divergence between the source and target domains. For reasonably similar domain pairs, the domain divergence is expected to be small which implies that a large number of target examples lies on the source side. We can formalize the label complexity reduction due to the domain separator hypothesis. As earlier, let $\mathcal{D}_s$ and $\mathcal{D}_t$ denote the source and target joint distributions, and let $p_{\mathcal{D}_s}(x)$ and $p_{\mathcal{D}_t}(x)$ be probabilities of an instance x belonging to the source and the target respectively, in the unlabeled pool used to train the domain separator hypothesis. Let $\varDelta$ denote the Mahalanobis distance between the source and target distributions. The Bayes error rate [15] of the domain separator hypothesis is: $E_{bayes} \leq \frac{2p_{\mathcal{D}_s}(x)p_{\mathcal{D}_t}(x)}{1+p_{\mathcal{D}_s}(x)p_{\mathcal{D}_t}(x)\varDelta}$. Thus, the label complexity reduction due to the domain separator hypothesis is proportional to the number of target examples misclassified by the domain separator hypothesis. This is again proportional to the Bayes error rate, which in turn is inversely related to the distance between the two domains.

2.4 Hybrid Oracle

ALDA additionally exploits the source domain information by using the source learned hypothesis (see, $\mathbf{w}_{src}$ in HYBRID of Fig. 1) as an oracle that provides labels for free. We denote this oracle by $\mathcal{O}_f$. Accordingly to the *Covariate Shift* assumption in domain adaptation, only the marginal distribution changes across domains whereas the conditional distribution remains fixed. If some target example appears to be close to the source domain then it is reasonable to assume that the prediction of the source classifer (which depends on the source conditional distribution) on that target sample should be close to the prediction of a *good* target classifier on that target sample. This explains the use of the source learned classifier as a free oracle for the target domain examples. Moreover, as in the standard active learning setting, we also have an expensive oracle $\mathcal{O}_c$. This leads to a hybrid setting which utilizes one of these two oracles for each actively sampled target example. The hybrid oracle starts with a domain adapted source initialized classifier ($\mathbf{u}_\phi$ in US of Fig. 1) and uses the domain separator hypothesis ($\mathbf{w}_{ds}$ in DS of Fig. 1) to assess which of the uncertain target examples lie on the source side and, for all such examples, it queries the labels from the free oracle $\mathcal{O}_f$. For the remaining uncertain examples that lie on the target side, the hybrid approach queries the expensive oracle $\mathcal{O}_c$. Although the oracle $\mathcal{O}_f$ is not perfect, the hope is that it can still provide correct labels for most of the target examples.

Algorithm 1 presents the final algorithm that combines all aforementioned schemes. This algorithm operates in a batch setting and we call it B-ALDA (for Batch-ALDA). As mentioned earlier (ref. Section 2.2), the importance ratio in line 2 of Algorithm 1 can be obtained by the techniques SCL [2], KMM [8], etc.

Algorithm 1. B-ALDA

```
input $L_s = \{\mathbf{x}_s, y\}$; $U_s$; $U_t$; maxCost (label budget $K$ and/or desired accuracy $\epsilon$);
output $\mathbf{v}$ (target classifier);
 1: cost := 0;
 2: $S := \tilde{L}_s$ (importance weighted $L_s$ learned using $L_s, U_s$ and $U_t$);
 3: $\mathbf{u}_{\varPhi}$ := learn a domain adapted source classifier using $S$;
 4: $\mathbf{w}_{ds}$ := learn a classifier using the data $\{U_s, +1\}$ and $\{U_t, -1\}$;
 5: $\mathbf{w}_{src}$ := learn a domain adapted source classifier using $L_s$;
 6: while (cost < maxCost) do
 7:   $\bar{\mathbf{x}}_t$ := US($\mathbf{u}_{\varPhi}$,$U_t$);          /* choose most informative target point */
 8:   $\hat{y}_{ds}$ := DS($\mathbf{w}_{ds}, \bar{\mathbf{x}}_t$);          /* compute source resemblance */
 9:   if ($\hat{y}_{ds} == +1$) then
10:     $y_t = \mathcal{O}_f(\mathbf{w}_{src}, \bar{\mathbf{x}}_t)$;          /* query the free oracle */
11:   else if ($\hat{y}_{ds} == -1$) then
12:     $y_t = \mathcal{O}_c(\bar{\mathbf{x}}_t)$;          /* query the costly oracle */
13:     cost ← cost + 1;
14:   end if
15:   $S = S \cup \{\bar{\mathbf{x}}_t, y_t\}$;
16:   retrain $\mathbf{u}_{\varPhi}$ using $S$;
17: end while
```

3 Online ALDA

In B-ALDA, the active learning module, at each iteration, chooses the data point that lies closest to the decision boundary. However, this approach is prohibitively slow for large or even moderately sized datasets. Hence, we propose Online ALDA (O-ALDA) which performs active learning in an online fashion and for each example decides whether or not to query its label. As in standard active learning, this query decision must be biased by the *informativeness* of the example.

To extend ALDA to the online setting, we adopt the label query strategy proposed in [3]. However, we note that our framework is sufficiently general and allows integration with other *active online sampling strategies*. The sampling scheme in [3] proceeds in rounds and at round i queries the label of the example x^i with probability $\frac{b}{b+|r^i|}$, where $|r^i|$ is the *confidence* (in terms of margin) of the current weight vector on x^i. Parameter b quantifies how aggressively the labels are being queried. A large value of b implies that, in expectation, a large number of labels will be queried (aggressive sampling) whereas a small value would lead to a small number of examples being queried (conservative sampling). For each label queried, the algorithm updates the current weight vector if the label was predicted incorrectly. It is easy to see that the total number of labels queried by this algorithm is $\sum_{i=1}^{T} \mathbb{E}[\frac{b}{b+|r^i|}]$, where T is the total number of rounds. At this point we note that the preprocessing stage of O-ALDA assumes the existence of some (maybe, a small amount) of target *unlabeled* data that can be utilized to construct the common representation. The online active learning in the target starts after this preprocessing phase when O-ALDA selectively queries the labels of the target data points that arrive in some random order.

Algorithm 2 presents the online variant of ALDA which we refer to as O-ALDA (for Online-ALDA). As shown in Theorem 1, our proposed O-ALDA yields provable guarantees on mistake bounds and label complexity.

Theorem 1. *Let $S = ((x_1, y_1), \ldots, (x_T, y_T)) \in (\mathbb{R}\times\{-1, +1\})^T$ be any sequence of examples and $\mathcal{UP}_T$ the (random) set of update trials for the algorithm (i.e., the set of trials $i \leq T$ such that $\hat{y}^i \neq y^i$ and $Z^i = 1$). Let $\boldsymbol{v}_0$ be the weight vector with which the base target classifier is initialized and r^i be the margin of* O-ALDA *on example x^i. Then the expected number of mistakes made by the algorithm is upper bounded by*

$$\inf_{\gamma>0} \inf_{\boldsymbol{v}^* \in \mathbb{R}^D} \left(\frac{(2b+1)}{2b} \mathbb{E}\left[\sum_{i \in \mathcal{UP}_T} \frac{1}{\gamma} D_\gamma(\boldsymbol{v}^*; (\hat{x}^i, y^i)) \right] + \frac{(2b+1)^2}{8b} \frac{||\boldsymbol{v}^* - \boldsymbol{v}_0||^2}{\gamma^2} \right)$$

The expected number of labels queried by the algorithm is equal to $\sum_{i=1}^{T} \mathbb{E}[\frac{b}{b+|r^i|}]$.

In the above theorem, γ refers to some margin greater than zero such that the cumulative hinge loss of the optimal target hypothesis $\mathbf{v}^*$ on S is given by $\sum_1^T D_\gamma(\mathbf{v}^*; (x^i, y^i))$, where $D_\gamma(\mathbf{v}^*; (x^i, y^i)) = max\{0, \gamma - y^i \mathbf{v}^{*T} x^i\}$ is the hinge-loss on example i. In Appendix A, we discuss the above theorem and provide a proof sketch for the mistake bound and the label complexity of O-ALDA. In

Algorithm 2. O-ALDA

```
input $b > 0$; $L_s = \{\mathbf{x}_s, y\}$; $U_s$; $U_t$; maxCost (label budget $K$/desired accuracy $\epsilon$);
output $\mathbf{v}$ (target classifier);
cost := 0;
$\mathbf{u}_\Phi$ := learn a domain adapted source classifier using $L_s, U_s$ and $U_t$;
$\mathbf{w}_{ds}$ := learn a classifier using the data $\{U_s, +1\}$ and $\{U_t, -1\}$;
$\mathbf{w}_{src}$ := learn a domain adapted source classifier using $L_s$;
while ( $(i <= T)$ & (cost < maxCost) ) do
    $r^i := \text{US}(\mathbf{u}_\Phi, \mathbf{x}_t^i)$;        /* compute margin of $i^{th}$ target point */
    $\hat{y}_{ds}^i := \text{DS}(\mathbf{w}_{ds}, \mathbf{x}_t^i)$;        /* compute source resemblance */
    sample $Z^i \sim \mathit{Bernoulli}(\frac{b}{b+|r^i|})$;
    if $(Z^i == 1)$ then
        if $(\hat{y}_{ds}^i == +1)$ then
            $y_t^i = \mathcal{O}_f(\mathbf{w}_{src}, \mathbf{x}_t^i)$;        /* query the free oracle */
        else if $(\hat{y}_{ds}^i == -1)$ then
            $y_t^i = \mathcal{O}_c(\mathbf{x}_t^i)$;        /* query the costly oracle */
            cost ← cost + 1;
        end if
        if $(y_t^i \neq \mathbf{u}_\Phi^T \mathbf{x}_t^i)$ then
            update $\mathbf{u}_\Phi$ using online update rule (such, as perceptron);
        end if
    end if
end while
```

addition, we discuss the conditions on $\mathbf{v}_0$ that lead to improved mistake bounds in domain adaptation settings as compared to the case where there is no access to data from a related source domain.

4 Experiments

In this section, we demonstrate the empirical performance of our algorithms and compare them with a number of baselines.

4.1 Setup

Datasets: We present results for **Sentiment** and **Landmine** datasets. The **Sentiment** dataset consists of user reviews of eight product types (apparel, books, DVD, electronics, kitchen, music, video, and other) from Amazon.com. The sentiment classification task for this dataset is binary classification which corresponds to classifying a review as positive or negative. The dataset consists of several domain pairs with varying $\mathcal{A}$-distances, akin to a sense described in [1]. Table 1 shows some of the domain pairs used in our experiments and their corresponding domain divergences in terms of the $\mathcal{A}$-distance [1].

To compute the $\mathcal{A}$-distance from finite samples of source and target domain, we use a surrogate to the true $\mathcal{A}$-distance (the *proxy* $\mathcal{A}$-distance) in a manner similar to [1]. First, we train a linear classifier to separate the *source* domain

Table 1. Proxy $\mathcal{A}$-distances between some domain pairs in the sentiment data

Source	Target	$\mathcal{A}$-distance
Dvd (D)	Books (B)	0.7616
Dvd (D)	Music (M)	0.7314
Books (B)	Apparel (A)	0.5970
Dvd (D)	Apparel (A)	0.5778
Electronics (E)	Apparel (A)	0.1717
Kitchen (K)	Apparel (A)	0.0459

from the *target* domain using only unlabeled examples from both. The average per-instance hinge-loss of this classifier subtracted from 1 serves as our estimate of the *proxy* $\mathcal{A}$-distance. A score of 1 means perfectly separable distributions whereas a score of 0 means that the two distributions are essentially the same. The amount of useful information that can be leveraged from the other domain would depend on how similar the two domains are. To this end, we therefore choose two datasets from the sentiment data - one with a domain-pair that is reasonably close (Kitchen→Apparel), and another with a domain-pair that is reasonably far apart (DVD→Books).

Our second dataset (**Landmine**) is the real Landmine Detection data [16] which consists of 29 datasets. The datasets 1 to 10 are collected at foliated regions whereas the datasets 20 to 24 are collected from bare earth or desert regions. We combined datasets $1 - 5$ as our source domain and treat dataset 24 as the target domain.

Methods: Table 2 summarizes the methods used with a brief description of each. Among the first three (ID, sDA, Feda), Feda [6] is a state-of-the-art *supervised* domain adaptation method but assumes *passively* acquired labels. The first three methods (ID, sDA, Feda) acquire labels *passively.* The last five (Alzi, Alri, Alsi, B-Alda and O-Alda) methods in Table 2 acquire labels in an active fashion. As the description denotes, Alzi and Alri start active learning in *target* with a zero initialized and randomly initialized hypothesis, respectively. It is also important to distinguish between Alsi and Alda (which jointly denotes both B-Alda and O-Alda). While both are products of our proposed Alda framework, Alsi uses an unmodified source classifier learned only from *source* labeled data as the initializer, whereas Alda (i.e., both B-Alda and O-Alda) uses an *unsupervised* domain-adaptation technique (i.e., without using labeled target data) to learn the initializer.

In our experiments, we use the instance reweighting approach [14] to construct the unsupervised domain adaptated classifier $\mathbf{u}_\phi$. However, we note that this step can also be performed using any other unsupervised domain adaptation technique such as Structural Correspondence Learning (SCL) [2] and Kernel Mean Matching (KMM) [8].

We compare all the approaches based on classification accuracies achieved for a fixed unlabeled pool of target examples with varying label budgets. For B-Alda, we use a margin based classifier (SVM) whereas for O-Alda we use vanilla

Table 2. Description of the methods compared

Method	**Summary**	**Active ?**
ID	In-domain data	No
sDA	Unsupervised domain adaptation followed by *passively* chosen labeled target data	No
Feda	Frustratingly Easy Domain Adaptation [6]	No
Alzi	Active learning zero initialized	Yes
Alri	Active learning random initialized (with fixed label budget)	Yes
Alsi	Active learning source (hypothesis) initialized	Yes
B-Alda	Batch active learning domain adapted	Yes
O-Alda	Online active learning domain adapted	Yes

Perceptron as the base classifier. All online experiments have been averaged over multiple runs with respect to random data order permutations.

4.2 B-Alda Results

We present results for B-Alda using a fixed target unlabeled pool and varying target label budgets. Since, domain adaptation is required only when there are small amounts of labeled data in the target, we limit our target label budget to values that are much smaller than the size of the unlabeled target data pool. In addition, due to long running times of our batch Alda (owing to repeated retraining), we report results on relatively smaller target pool sizes. The B-Alda results are presented for a unlabeled target pool size of 2500 data points.

Table 3. Classification accuracies and number of labels requested. Note: ID, sDA and Feda are given labels of all examples in the target pool.

(a) DVD→Books

Method	**Target Label Budget**				
	100	**200**	**300**	**400**	**500**
	Acc	**Acc**	**Acc**	**Acc**	**Acc**
ID	50.83	57.86	62.42	55.69	62.68
sDA	62.18	62.78	55.75	52.45	50.49
Feda	63.92	64.27	64.88	65.94	66.19
Alzi	54.40	54.36	54.33	54.33	54.33
Alri	54.99	59.42	61.28	65.81	65.52
Alsi	**63.75**	**66.26**	**68.73**	63.10	62.08
B-Alda	63.40	65.17	67.84	**68.61**	**68.51**
Acc: Accuracy					

(b) Kitchen→Apparel

Method	**Target Label Budget**				
	100	**200**	**300**	**400**	**500**
	Acc	**Acc**	**Acc**	**Acc**	**Acc**
ID	48.40	43.44	44.92	48.40	49.77
sDA	52.78	55.41	57.37	53.60	46.37
Feda	70.47	69.97	70.06	71.83	69.96
Alzi	54.56	54.50	54.44	54.44	54.44
Alri	64.97	66.86	69.01	70.40	71.06
Alsi	**74.91**	70.58	**72.97**	**72.34**	72.29
B-Alda	71.30	**70.90**	71.19	71.73	**73.07**
Acc: Accuracy					

Sentiment Classification: Table 3a and Table 3b present the results for the domain pairs DVD→Books and Kitchen→Apparel, respectively. For these domain pairs, both Alsi and B-Alda substantially outperform all other baselines. For the distant source-target pair (DVD→Books), Alsi performs very well for a small number of target labels (say, 100 and 200). As the number of target labels increases B-Alda consistently improves with increasing number of target labels and finally outperforms Alsi. When the source-target pairs are reasonably close (Kitchen→Apparel), both Alsi and B-Alda have similar prediction accuracies which are in turn are much higher that the baseline accuracies.

Table 4. AUC scores and labels requested for the **Landmine** dataset

Method	**Target Budget (300)**
	AUC
ID	0.59
sDA	0.60
Feda	0.56
Alzi	0.59
Alri	0.53
Alsi	0.63
B-Alda	**0.65**
AUC: AUC score **Lab:** Labels Requested	

__Landmine Detection:__ The **Landmine** dataset has a high class imbalance (only about 5% positive examples), so we report AUC (area under the ROC curve) scores instead of accuracies. We compare our algorithms with other baselines in terms of the AUC score on the entire pool of target data (the pool size was 300; rest of the examples in dataset 24 were treated as test data). As shown in Table 4, our approaches perform better than the other baselines with the domain separator based B-Alda doing the best (in terms of AUC scores).

We do not report any label complexity result for B-Alda as the nature of the algorithm is such that it iterates until the entire label budget is exhausted. Hence, in all the results presented above in Table 3a, Table 3b and Table 4, the number of labels used is equal to the target label budget provided.

4.3 O-Alda Results

One of the goals to propose an online variant for Alda is to make the proposed approach scale efficiently for larger target pool sizes because batch mode Alda requires repeated retraining. On the other hand, an online active learner is an efficient alternative because it allows incremental update of the learner for each new selected data point. In this section, we present results for O-Alda and demonstrate the scalability of the Alda framework to larger target pool sizes. The results for O-Alda use the entire target unlabeled pool ($\sim$ 7000 for **Sentiment** data). As a result, the label budget allocated is also much larger as compared to B-Alda. We note that ID and sDA and Feda have been made online by the use of the perceptron classifier. In addition, the same online active strategy as O-Alda has been used for Alzi, Alri and Alsi.

__Sentiment Classification:__ The results are shown in Table 5a and Table 5b. As the results indicate, on both datasets, our approaches (Alsi and Alda) perform consistently better than the baseline approaches (Table 2) which also include one of the state-of-the-art supervised domain adaptation algorithms (Feda). We note that Alda outperforms Alsi for Kitchen→Apparel as compared to DVD→Books. When the domains are far (DVD→Books), the performance of

Table 5. Classification accuracies and number of labels requested. Results are averaged over 20 runs (w.r.t. different permutations of the training data). Note: ID, sDA and Feda are given labels of all examples in the target pool.

(a) DVD→Books

Met-hod	**Target Label Budget**				
	1000	**2000**	**3000**	**4000**	**5000**
	Acc(±Std)	**Acc(±Std)**	**Acc(±Std)**	**Acc(±Std)**	**Acc(±Std)**
ID	65.94(±3.40)	66.66(±3.01)	67.00(±2.40)	65.72(±3.98)	66.25(±3.18)
sDA	66.17(±2.57)	66.45(±2.88)	65.31(±3.13)	66.33(±3.51)	66.22(±3.05)
Feda	67.31(±3.36)	68.47(±3.15)	68.37(±2.72)	66.95(3.11)	67.13(±3.16)
Alzi	66.24(±3.16)	66.72(±3.30)	63.97(±4.82)	66.28(±3.61)	66.36(±2.82)
Alri	51.79(±4.36)	53.12(±4.65)	55.01(±4.20)	57.56(±4.18)	58.57(±2.44)
Alsi	**68.22(±2.17)**	**69.65(±1.20)**	**69.95(±1.55)**	70.54(±1.42)	**70.97(±0.97)**
O-Alda	67.64(±2.35)	68.89(±1.37)	69.49(±1.63)	**70.55(1.15)**	70.65(±0.94)
Acc: Accuracy | **Std:** Standard Deviation					

(b) Kitchen→Apparel

Met-hod	**Target Label Budget**				
	1000	**2000**	**3000**	**4000**	**5000**
	Acc(±Std)	**Acc(±Std)**	**Acc(±Std)**	**Acc(±Std)**	**Acc(±Std)**
ID	69.64(±3.14)	69.61(±3.17)	69.36(±3.14)	69.77(±3.58)	70.77(±3.05)
sDA	69.70(±2.57)	70.48(±3.42)	70.29(±2.56)	70.86(±3.16)	70.71(±3.65)
Feda	70.05(±2.47)	69.34(±3.50)	71.22(±3.00)	71.67(±2.59)	70.80(±3.89)
Alzi	70.09(±3.74)	69.96(±3.27)	68.6 (±3.94)	70.06(±2.84)	69.75(±3.26)
Alri	52.13(±5.44)	56.83(±5.36)	58.09(±4.09)	59.82(±4.16)	62.03(±2.52)
Alsi	73.82(±1.47)	**74.45(±1.27)**	75.11(±0.98)	75.35(±1.30)	**75.58(±0.85)**
O-Alda	**73.93(±1.84)**	74.18(±1.85)	**75.13(±1.18)**	**75.88(±1.32)**	**75.58(±0.97)**
Acc: Accuracy | **Std:** Standard Deviation					

Alda depends on the underlying domain adaptation technique. However, when the domains are close (Kitchen→Apparel), Alda performs better than Alsi. This behavior suggests that the performance gains achieved by these variants is significant when the source and target domains are *reasonably close.*

Landmine Detection: Similar to B-Alda results, in this case also we used the entire pool of 300 target data points. The rest of the examples in dataset 24 were treated as test data. As earlier, our approaches perform better than the other baselines with the domain separator based O-Alda demonstrating slightly better AUC score and slightly lesser label complexity as compared to online Alsi. Table 6 presents the AUC scores and the label complexities of the various methods.

4.4 Remarks

For all datasets considered, both batch and online versions of Alda demonstrate substantial improvement of prediction accuracy for **Sentiment** data

(∼ (**0.4%** − **5.09%**)). This improvement is particularly high when the domains are reasonably similar (for example, Kitchen→Apparel in Table 3b and Table 5b). In addition, the **Landmine** data reports AUC scores (not accuracies), and 1% increase in AUC score implies substantial improvement.

Table 6. AUC scores and labels requested for the **Landmine** dataset. Results are averaged over 20 runs.

Method	**Target Budget (300)**
	AUC±Std (Lab)
ID	0.57±0.03 (-)
sDA	0.60±0.02 (-)
Feda	0.52±0.04 (-)
Alzi	0.61±0.02 (284)
Alri	0.56±0.05 (229)
Alsi	0.65±0.02 (244)
O-Alda	**0.67±0.03 (241)**
AUC: AUC score	
Std: Standard Deviation	
Lab: Labels Requested	

For **Sentiment** and **Landmine** datasets, both Alsi and Alda (i.e., B-Alda and O-Alda) demonstrate improvement in *prediction accuracy for a fixed label budget* when compared to other baselines. Apart from the results for DVD→Books in the batch setting (Table 3a), the prediction accuracies obtained by Alsi and Alda in all other cases are comparable. However, to get a better sense of the robustness of these two approaches, we compare the number of mistakes made by the online variants of these two approaches during the training phase. Table 7 presents the results for **Sentiment** data. As can be seen, in almost all case the number of mistakes made by O-Alda is much lesser (almost half in many cases) than online Alsi. Hence, irrespective of the nearness or farness of the source-target domain pairs, Alda is a better choice as compared to Alsi.

Table 7. Number of mistakes made by Alsi and O-Alda for **Sentiment** data

	Target Label Budget									
	1000	**2000**	**3000**	**4000**	**5000**	**1000**	**2000**	**3000**	**4000**	**5000**
	Number of Mistakes									
Method	DVD→Books					Kitchen→Apparel				
Alsi	**369**	**739**	1117	1460	1816	245	532	810	1097	1088
O-Alda	384	741	**1000**	**1012**	**1004**	**232**	**478**	**549**	**551**	**556**

5 Related Work

Active learning in a domain adaptation setting has received little attention so far and, to the best of our knowledge, there exists no prior work that presents a principled framework to harness domain adaptation for active learning. One interesting setting was proposed in [4] where the authors apply active learning for word sense disambiguation in a domain adaptation setting. In addition, they also improve vanilla active learning when combined with domain adaptation. However, their approach does not use the notions of domain separator and hybrid oracle. Moreover, unlike our approach, their method only works in a batch setting.

Active learning in an online setting has been discussed in [5] and [3]. The work of [5] assumes input data points uniformly distributed over the surface of an unit sphere. However, we cannot make such distributional assumptions for domain adaptation. As mentioned earlier, [3] provide worst-case analysis which is independent of any input data distribution. However, none of these explicitly consider the case of domain adaptation. Nonetheless, the framework of [3] folds nicely into our proposed ALDA framework. [10] present extensive empirical results to compare the performance of the two aforementioned approaches. However, all these settings are different from our in that these works consider only active learning in an online setting without leveraging inter-domain information.

A combination of transfer learning with active learning has been presented in [13]. One drawback of their approach is the requirement of an initial pool of labeled target domain data which helps train the in-domain classifier. Without this in-domain classifier, no transfer learning is possible in their setting.

6 Discussions and Future Work

In this work, we have considered a domain adaptation setting, and presented a framework that helps leverage inter-domain information transfer while performing active learning in the target. Both the batch and online versions of the proposed ALDA empirically demonstrate the benefits of domain transfer for active learning.

At present, ALDA is oblivious to the feature set used and, as such, does not depend on domain knowledge and feature selection. It takes all features into consideration. Nonetheless, it is possible that in the feature space, not all features contribute equally while transferring information from source to target and without a priori information about the source and target domains, it is difficult to assess which features might maximally benefit the transfer of parameters from source to target. However, if prior domain knowledge about the target is available from related source domains, then one can potentially leverage active learning to selectively choose *only* those features that transfer maximum information between the two domains.

An alternative approach to leverage feature information for ALDA is to perform active learning on features. There exists work in active learning that queries labels for features [7] and, in some cases, queries labels for both instances and features in tandem [11]. We note that this is different from the above where active learning can essentially be used as a tool for feature selection. In this case, active strategies query labels that exploit both instance and feature informativeness (for e.g., in NLP, consider querying labels for rare words which serve as informative features in the target domain). It would be interesting to extend the proposed ALDA to perform active domain transfer by querying labels of both instances and features.

Acknowledgements. This work was sponsored in part by the NSF grants CCF-0953066 and CCF-0841185 and in part by the Consortium for Healthcare

Informatics Research (CHIR), VA HSR HIR 08-374, VA Informatics and Computing Infrastructure (VINCI) and VA HSR HIR 08-204. This work was also partially supported by the NSF grant IIS-0712764. The authors gratefully acknowledge the support of the grants. Any opinions, findings, and conclusion or recommendation expressed in this material are those of the author(s) and do not necessarily reflect the view of the funding agencies or the U.S. government.

References

1. Ben-David, S., Blitzer, J., Crammer, K., Pereira, F.: Analysis of representations for domain adaptation. In: NIPS 2006, Vancouver, Canada (December 2006)
2. Blitzer, J., Mcdonald, R., Pereira, F.: Domain adaptation with structural correspondence learning. In: EMNLP 2006, Sydney, Australia (July 2006)
3. Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Worst-case analysis of selective sampling for linear classification. JMLR 7 (2006)
4. Chan, Y.S., Ng, H.T.: Domain adaptation with active learning for word sense disambiguation. In: ACL 2007, Prague, Czech Republic (June 2007)
5. Dasgupta, S., Kalai, A.T., Monteleoni, C.: Analysis of perceptron-based active learning. JMLR 10 (2009)
6. Daumé III, H.: Frustratingly easy domain adaptation. In: ACL 2007, Prague, Czech Republic (June 2007)
7. Druck, G., Settles, B., McCallum, A.: Active learning by labeling features. In: EMNLP 2009, Singapore (August 2009)
8. Huang, J., Smola, A.J., Gretton, A., Borgwardt, K.M., Schölkopf, B.: Correcting sample selection bias by unlabeled data. In: NIPS 2007, Vancouver, Canada (2007)
9. Jiang, J.: A literature survey on domain adaptation of statistical classifiers (2008)
10. Monteleoni, C., Kääriäinen, M.: Practical online active learning for classification. In: IEEE CVPR Workshop on Online Learning for Classification, Minneapolis, USA (June 2007)
11. Raghavan, H., Madani, O., Jones, R.: Active learning with feedback on features and instances. JMLR 7 (December 2006)
12. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison (2009)
13. Shi, X., Fan, W., Ren, J.: Actively transfer domain knowledge. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 342–357. Springer, Heidelberg (2008)
14. Sugiyama, M., Nakajima, S., Kashima, H., von Bünau, P., Kawanabe, M.: Direct importance estimation with model selection and its application to covariate shift adaptation. In: NIPS 2007, Vancouver, Canada (December 2007)
15. Tumer, K., Ghosh, J.: Estimating the bayes error rate through classifier combining. In: ICPR 1996, Vienna, Austria, vol. 2 (August 1996)
16. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with dirichlet process priors. JMLR 8 (2007)

A Discussion of Theorem 1

To conserve space, we skip presenting a detailed proof of the mistake bound in Theorem 1. Proceeding in a manner similar to the proof of Theorem 1 of

[3], it can be seen that almost all terms in the final expression for the mistake bound cancel out by the telescopic argument. The term that remains is $||\mathbf{v}^* - \mathbf{v}_0||^2$ and the mistake bound follows. It is easy to see that setting $\mathbf{v}_0 = 0$ in Theorem 1 yields mistake bounds for online active learning in traditional single task settings. We note that, the first term in the mistake bound of Theorem 1 is the cumulative hinge loss of the *optimal* target classifier which is the same for both domain adaptation and non-domain adaptation (traditional single task) settings and hence is independent of the initialization used. The second term in the mistake bound, in our case, is smaller than single task settings provided $\theta \leq cos^{-1}\left(\frac{||\mathbf{v}_0||}{2||\mathbf{v}^*||}\right)$, where θ is the angle between the initializing hypothesis $\mathbf{v}_0$ and the target hypothesis $\mathbf{v}^*$. Without loss of generality, assuming the norm of $\mathbf{v}_0$ and $\mathbf{v}^*$ stays fixed (which is true since both the *initial* and the *optimal* hypotheses remain unchanged during learning in target domain), as the value of θ decreases, it causes $||\mathbf{v}^* - \mathbf{v}_0||^2$ to decrease, leading to our claim of reduced mistake bounds. Thus, in our framework, θ incorporates the notion of the domain separation that improves the mistake bounds. For small values of θ, the source and target domains have high proximity such that the *initial target* hypothesis $\mathbf{v}_0$ lies reasonably close to the *optimal target* hypothesis $\mathbf{v}^*$. As a result, is such cases, O-Alda is expected to make a smaller number of mistakes to get to the optimal hypothesis.

Now, we present an intuitive argument for the lower label complexity of O-Alda as compared to single task online active settings. O-Alda is initialized with a *non-zero hypothesis* $\mathbf{v}_0 = \mathbf{w}_{src}$ learned using data from a related source domain. Hence, the sequence of hypotheses O-Alda produces will in expectation have higher confidences margins $|\bar{r}^i|$ as compared to some *zero initialized hypothesis*. Therefore, at each step the sampling probability of O-Alda given by $\frac{b}{b+|\bar{r}^i|}$ will also be smaller, which will lead to a smaller number of queried labels since it is nothing but $\sum_{i=1}^{T} \mathbb{E}[\frac{b}{b+|\bar{r}^i|}]$.

Efficiently Approximating Markov Tree Bagging for High-Dimensional Density Estimation

François Schnitzler[1], Sourour Ammar[2], Philippe Leray[2],
Pierre Geurts[1], and Louis Wehenkel[1]

[1] Université de Liège, Department of EECS and GIGA-Research,
Grande Traverse, 10 - B-4000 Liège - Belgium
{fschnitzler,P.Geurts,L.Wehenkel}@ulg.ac.be

[2] Ecole Polytechnique de l'Université de Nantes, Knowledge and Decision Team,
Laboratoire d'Informatique de Nantes Atlantique (LINA) UMR 6241, France
{sourour.ammar,philippe.leray}@univ-nantes.fr

Abstract. We consider algorithms for generating *Mixtures of Bagged Markov Trees*, for density estimation. In problems defined over many variables and when few observations are available, those mixtures generally outperform a single Markov tree maximizing the data likelihood, but are far more expensive to compute. In this paper, we describe new algorithms for approximating such models, with the aim of *speeding up learning without sacrificing accuracy*. More specifically, we propose to use a filtering step obtained as a by-product from computing a first Markov tree, so as to avoid considering poor candidate edges in the subsequently generated trees. We compare these algorithms (on synthetic data sets) to Mixtures of Bagged Markov Trees, as well as to a single Markov tree derived by the classical Chow-Liu algorithm and to a recently proposed randomized scheme used for building tree mixtures.

Keywords: mixture models, Markov trees, bagging, randomization.

1 Introduction

Estimation of multivariate probability densities from observational data is a widely used strategy to tackle decision making problems under uncertainty. A density model can be used to answer various queries about the underlying data generation mechanism (also called performing inference), such as computing the likelihood of observing a problem instance, or estimating the conditional probability density of a subset of variables given observed values of another subset.

The framework of probabilistic graphical models [18,28] provides well founded approaches to model probability densities and to perform inference by combining graph theory, with statistics and algorithmics. The structure of a graphical model encodes relationships between variables while its parameters quantify those interactions. Bayesian networks are a class of models that encode a joint probability density over a set of variables by a product of conditional probability densities (see Sect. 2). Both learning and inference are however NP-hard with those models when the underlying graph is unconstrained [11,22].

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 113–128, 2011.

To cope with the problem size expansion faced today in many applications due to the rapid increase in measurement resolution, many learning methods for bayesian networks incorporate some constraints on their graphical structure, e.g. [7,14,16,31]. In that regard, an interesting subset of those models is the class of Markov trees, where each conditional probability distribution is conditioned on a single variable (except the root) [28]: learning a Markov tree maximizing the data likelihood by the Chow-Liu algorithm [10] has a computational complexity essentially quadratic in the number of variables, while performing inference with such models is of linear complexity. Another advantage of Markov trees is their small number of parameters, which reduces the risk of overfitting when data is scarce. However, for problems with very large numbers of variables and low sample size this model class may already be too large, and it may be desirable to impose additional regularization constraints on top of this method [23].

Bootstrap aggregation (bagging) [8,12] is a meta-algorithm that compensates for a lack of data by applying a given algorithm on several bootstrap replicas of the original data set and averaging the predictions of the resulting models. A bootstrap replica is obtained by randomly drawing (with replacement) original samples and copying them into the replica. Averaging the predictions from an ensemble of models derived from an ensemble of bootstrap replicas leads to a decrease in variance and hence a reduction in overfitting. This meta-algorithm, originally developed in the context of supervised learning, has already been applied for learning probabilistic graphical models e.g. [13,15], often to get a more robust structure but without consideration for inference on the said structure.

When one is willing to use bootstrapping to obtain a density model on which inference is tractable, one interesting possibility is to use bagged mixtures of Chow-Liu trees. Indeed, these have been shown to outperform single Chow-Liu trees, specially on high-dimensional problems with small sample sizes [2]. However, the extra computational cost with respect to learning one single Chow-Liu tree may prove problematic on very large problems. In this work we therefore investigate means to reduce this complexity by approximating the original bootstrap procedure. We propose to couple a first application of the Chow-Liu algorithm to either subsampling the set of candidate edges, or to a statistical test to detect irrelevant edges, in order to avoid considering all candidate edges in the subsequent runs of the Chow-Liu algorithm on subsequent bootstrap replicas. The second approach can be seen as applying a structural regularization to identify a skeleton comprising only potentially relevant edges, and then restricting the search of optimal Markov trees on subsequent bootstrap replicas within that smaller envelope instead of the complete set of all possible edges; it may hence also be beneficial in terms of accuracy in very small sample size conditions.

In Sect. 2 we cover the concept of bayesian networks, bayesian learning and mixtures of Markov trees in more details. We then describe the baseline algorithms for Markov tree based density models upon which we propose improvements (Sect. 3), before detailing our new algorithms (Sect. 4). The experiments performed to compare them in terms of accuracy, convergence speed and computing times are presented and discussed in Sect. 5.

2 Graphical Probability Density Model Learning

A bayesian network [28] is a probabilistic graphical model that encodes a joint probability density over a finite set $\mathcal{X}$ of n variables $\{X_1, X_2, ..., X_n\}$. Those variables correspond to the nodes of a Directed Acyclic Graph (DAG) $\mathcal{G}$ that encodes conditional independence relationships between variables and allows their algorithmic verification. The graphical structure actually defines a factorization of the joint density as a product of conditional densities of each variable X_i conditionally to the set of its parents $Pa_{\mathcal{G}}(X_i)$ in the graph:

$$P(\mathcal{X}) = \prod_{i=1}^{n} P(X_i | Pa_{\mathcal{G}}(X_i)) \ . \tag{1}$$

In the case of discrete variables, learning the parameters defining those conditional densities from data is relatively straightforward, but structure learning is not. There are three main structure learning approaches: a score-based, a constraint-based and a bayesian one [18].

In the *score-based approach* [5], a numerical criterion (maximum likelihood, BIC, AIC...) is defined over the set of DAGs, and learning can be defined as selecting, among all DAGs, the one that maximizes this score with respect to the data set. However, the number of possible DAGs (as well as the number of their equivalence classes) grows superexponentially with the number of variables n [29]. Since existing unconstrained score optimization algorithms are not scalable, simplifications must in practice be used. These may be achieved by reducing the number of candidate structures, either by restricting the resolution of the search space [6] or by limiting its range (e.g. by constraining the number of candidate parents or the global structures searched [14]).

The *constraint-based approach* [1] consists in extracting from the observational data a set of conditional independence relationships (statements $\mathcal{S}_i \perp \mathcal{S}_j | \mathcal{S}_k$, where $\mathcal{S}_i, \mathcal{S}_j, \mathcal{S}_k$ are disjoint subsets of variables), and searching for a structure that best matches those constraints. Algorithms typically consider the assessment of a polynomial number of independence relationships, and achieve this by limiting the cardinalities of the subsets of variables inspected.

The *bayesian averaging approach* [24] considers the set of all possible structures rather than identifying a single best one, and averages predictions from those structures in accordance with the goal of the learning procedure. Taking into account all possible structures is rarely possible, and approximations must thus be employed. One of these strategies is the bootstrap aggregation approach that we are considering in this article (see Sect. 2.1).

But the cost of inference must also be considered. Its complexity grows with the tree-width of the DAG [22], which is the minimum size, minus one, of the largest connected subgraph in a moralized and triangularized version of the DAG (obtained by first joining all non-adjacent parents of any variable, and then by chordalizing all cycles). Although many heuristic approaches to inference have been developed, many learning methods target low tree-width structures [7,14,16,31] and thus also limit inference complexity.

Markov trees, a subclass of bayesian networks, allow for scalable learning and inference; it consists of all bayesian networks where each variable has a single parent. The Chow-Liu algorithm (Sect. 3.1) produces a Markov tree maximizing the likelihood of a data set, and its complexity is essentially quadratic in the number of variables. The tree-width of a Markov tree is always one, and inference is thus linear in the number of variables. Both properties make Markov trees extremely interesting for high-dimensional density modeling.

2.1 Bagging in the Context of Learning Bayesian Networks

Bagging is a model averaging method where a given learning algorithm is randomized by applying it on m different bootstrap replica data sets, therefore resulting in m different models. A bootstrap replica $\mathbf{D}'$ of size p' is obtained from an original data set $\mathbf{D}$ of p observations by uniformly and independently drawing p' natural numbers $r_i \in [1, p]$, and by compiling $\mathbf{D}'$ by

$$\mathbf{D}'[i] = \mathbf{D}[r_i] \qquad\qquad \forall i \in [1, p'] \ , \tag{2}$$

where $\mathbf{D}[j]$ (resp. $\mathbf{D}'[k]$) refers to jth (resp. kth) observation of $\mathbf{D}$ (resp. $\mathbf{D}'$), and where typically (as in this paper) $p = p'$.

The result of the bagging algorithm is an average between the m models learnt from the m bootstrap replicas that typically exhibits a lower variance than a model learned directly from the original data set. This approach has been quite popular and effective in the context of supervised learning [8].

Bagging has been proposed for Gaussian density modeling [27], and for structure learning of graphical models, e.g. by considering the frequency of occurrence of interesting graphical features among the structures derived from bootstrap replicas [15], and also to improve score-based structure learning by incorporating the bootstrap procedure in the computation of the score [13]. Recently, it was proposed for generating ensembles of bagged Chow-Liu trees [2]; this latter approach is denoted in the rest of this paper by *Bagged Mixture of Chow-Liu Trees*; it is discussed more in detail in the next section.

2.2 Mixtures of Markov Trees

A mixture of Markov trees over a set of n variables $\mathcal{X}$ is a convex combination of a set $\hat{\mathcal{T}} = \{T_1, \ldots, T_m\}$ of m elementary Markov tree densities, i.e.

$$P_{\hat{\mathcal{T}}}(\mathcal{X}) = \sum_{i=1}^{m} \mu_i P_{T_i}(\mathcal{X}) \ , \tag{3}$$

where $\{\mu_i\}_{i=1}^{m}$ are the weights of the mixture ($\mu_i \in [0, 1]$ and $\sum_{i=1}^{m} \mu_i = 1$). The complexity of inference in this model is thus equal to m times the complexity of inference with a single tree, which is linear in the number n of variables.

Several learning algorithms of mixtures of Markov trees have already been proposed; they can be categorized into two groups: the maximum likelihood and the randomization approaches.

In the former approach, the mixture of trees is primarily used as a mean to exploit the good algorithmic properties of trees while improving their modeling capabilities. These methods include using the EM algorithm to partition the data between a given number of terms [25], or using clever reweighting schemes on the whole data set to fit modes of the density [21].

The second approach can be viewed as an attempt to approximate true bayesian learning in the space of Markov tree structures. In these methods, a set of tree models are generated using a more or less strongly randomized procedure, that can range from completely random structures based on Prüfer lists to bagged Mixtures of trees. The weights associated to these trees can be either uniform or proportional to the score of the structure based on the data set. A comparison of theses approaches can be found in [3]. The present work adopts this strategy and some of those methods are further described in Sect. 3.

An approach at the intersection of those two categories has been proposed in [17], where a MCMC exploration scheme is defined on the space of mixtures of trees using a Dirichlet process and a suitable prior on tree structures [26].

3 Baseline Markov Tree Based Learning Algorithms

In this section, we describe the three baseline methods of density estimation with Markov trees reused in this paper, and we state their computational complexity.

3.1 The Chow-Liu Algorithm for Learning a Markov Tree

The algorithm for learning a Markov tree structure $T_{CL}(\mathbf{D})$ maximizing the likelihood of a training set $\mathbf{D}$ was introduced by Chow and Liu [10]. It solves the optimization problem

$$T_{CL}(\mathbf{D}) = \arg\max_T \sum_{(X_i,X_j)\in\mathcal{E}(T)} I_{\mathbf{D}}(X_i;X_j) \ , \tag{4}$$

where $\mathcal{E}(T)$ is the set of edges in T, constrained to be a tree, and where $I_{\mathbf{D}}(X_i,X_j)$ is the maximum likelihood estimate of the mutual information among variables X_i and X_j computed from the dataset $\mathbf{D}$ (composed of p observations).

Algorithm 1 has two steps: first $I_{\mathbf{D}}(X_i,X_j)$ $(\forall i = 1 \dots n, \forall j = i+1 \dots n)$ are computed to fill an $n \times n$ symmetrical matrix (MI), then used to compute a maximum weight spanning tree (MWST, e.g. by [19] as here, or by [9]).

Algorithm 1 (Chow-Liu (CL) tree)

```
1. $MI = [0]_{n\times n}$
2. Repeat for $i_1 = 1, \cdots, n$:
       Repeat for $i_2 = i_1 + 1, \cdots, n$:
       $MI[i_1,i_2] = MI[i_2,i_1] = I_{\mathbf{D}}(X_{i_1};X_{i_2})$
3. $T_{CL} = \mathbf{MWST}(MI)$
4. Return $T_{CL}$.
```

Step 2 requires $\mathcal{O}(n^2p)$ computations, while computing a MWST has a complexity of $E\log(E)$ with E the number of candidate edges. Here $E = n(n-1)/2$, so that, for fixed sample size p, the complexity is $\mathcal{O}(n^2\log(n^2)) \equiv \mathcal{O}(n^2\log(n))$.

3.2 Bagging of Chow-Liu Markov Trees

Bagging of the Chow-Liu algorithm is described by Algorithm 2.

Algorithm 2 (Generating a mixture of bagged Chow-Liu trees)

```
1. $\hat{\mathcal{T}} = \{\}$
2. Repeat for $j = 1, \cdots, m$:
   (a) $\mathbf{D}'$= bootstrap($\mathbf{D}$)
   (b) $\hat{\mathcal{T}} = \hat{\mathcal{T}} \cup \{\textbf{Chow-Liu}(\mathbf{D}')\}$
3. Return $\hat{\mathcal{T}}$, $\mu = \{1/m, \cdots, 1/m\}$.
```

The complexity of Algorithm 2 is m times the complexity of the Chow-Liu algorithm, or $\mathcal{O}(mn^2 \log(n))$, for fixed sample size p.

Notice that it was shown in [30] that learning the parameters of each tree in $\hat{\mathcal{T}}$ on $\mathbf{D}$ rather than on the replica $\mathbf{D}'$ used to generate its structure improves accuracy, and we will therefore use $\mathbf{D}$ to estimate the parameters of all Markov trees generated by all the algorithms studied in this paper, according to [30].

3.3 Inertial Search Heuristic

This algorithm [4] improves the computational complexity of the Bagging method by limiting to a specified number K the number of variable pairs and mutual informations computed and considered for each MWST construction. Constructing $\hat{\mathcal{T}}$ is done here by a sequential procedure: for optimizing the first tree, a random subset of K edges is considered, and then for each subsequent tree T_i, the considered subset is initialized by the edges of the previous tree, $\mathcal{S} = \mathcal{E}(T_{i-1})$, and completed with an additional random subset of $K - |\mathcal{E}(T_{i-1})|$ edges.

Algorithm 3 (Inertial search of mixtures of Markov trees (ISH))

```
1. $\hat{\mathcal{T}} = \{\}$, $\mathcal{S} = \{\}$
2. Repeat for $j = 1, \cdots, m$:
   (a) $\mathbf{D}'$= bootstrap($\mathbf{D}$)
   (b) $MI = [0]_{n \times n}$
   (c) Repeat for $k = 1, \cdots, |\mathcal{S}|$:
       i. $(i_1, i_2) = GetIndices(\mathcal{S}[k])$
       ii. $MI[i_1, i_2] = MI[i_2, i_1] = I_{\mathbf{D}'}(X_{i_1}; X_{i_2})$
   (d) Repeat for $k = |\mathcal{E}| + 1, \cdots, K$
       i. $(i_1, i_2)$ = drawNewRandomEdge
       ii. $MI[i_1, i_2] = MI[i_2, i_1] = I_{\mathbf{D}'}(X_{i_1}; X_{i_2})$
   (e) $T$= MWST(MI)
   (f) $\mathcal{S} = \mathcal{E}(T)$
   (g) $\hat{\mathcal{T}} = \hat{\mathcal{T}} \cup \{T\}$
3. Return $\hat{\mathcal{T}}$, $\mu = \{1/m, \cdots, 1/m\}$.
```

The parameter K controls the computational complexity of the method, which is $\mathcal{O}(mK \log K)$. We use $K = Cn \ln n$ as in [4] (with $C = 1$ in most of our simulations) leading to a complexity approximately of $\mathcal{O}(mn \log(n))$.

4 Proposed Algorithms

In this section we propose our alternative algorithms. They all start by computing a Chow-Liu tree on the original data set $\mathbf{D}$ and they then use the results of this computation for accelerating the generation of subsequent ensemble terms.

4.1 Improving the Inertial Search Heuristic by Warm Start

While Algorithm 3 is of log-linear complexity in n and gradually improves as new trees are added to the model, it consists essentially in an exploration of the matrix MI of mutual informations. Notice that without bagging (i.e. by using $\mathbf{D'} = \mathbf{D}$ at all iterations), this algorithm would eventually converge to the Chow-Liu tree, since Tarjan's red rule [32] implies that the lightest edge of any cycle is not part of the MWST. However, the number of iterations needed to fully explore the matrix essentially increases with n, since

$$\lim_{n\to\infty} \frac{\text{Edges considered at each iteration}}{\text{Total edges}} = \frac{\mathcal{O}(n\log n)}{\mathcal{O}(n^2)} = 0 \ , \tag{5}$$

and hence the algorithm will take longer and longer to converge as n increases (see also our experimental results in the next section).

We hence modified this method, by changing the first iteration so as to start with a more optimal set of edges ($\mathcal{E}(T_{CL})$ computed by the Chow-Liu algorithm based on a complete matrix of mutual informations; see Algorithm 4).

Algorithm 4 (Warm start inertial research procedure (Warm start ISH))

```
1. $\hat{\mathcal{T}} = \{\mathbf{Chow\text{-}Liu}(\mathbf{D})\}$, $\mathcal{S} = \mathcal{E}(\mathbf{Chow\text{-}Liu}(\mathbf{D}))$
2. Repeat for $j = 2, \cdots, m$:
   $\cdots$ (identical to points (a) to (g) of Algorithm 3)
3. Return $\hat{\mathcal{T}}$, $\mu = \{1/m, \cdots, 1/m\}$.
```

The complexity of this method is $\mathcal{O}(n^2 \log(n) + mK \log(K))$ where K is the number of edges considered at each iteration after the first one. As in Algorithm 3, we set $K = Cn \ln n$. In practice the gain in convergence speed strongly compensates for the increased complexity needed for computing the first term (see our results in the next section).

Alternatively, both methods could be viewed as a stochastic walk in the space of Markov tree structures that at convergence will attain the set of good structures. Algorithm 3 however starts very far from this set while the variant we propose in Algorithm 4 starts from a more sensible initial guess.

4.2 Pruned Mixtures of Bagged Chow-Liu Trees

The Chow-Liu method (Algorithm 1), and its bagging (Algorithm 2) compute connected Markov tree structures of maximum likelihood over the data sets they get as input. But in high-dimensional problems (with $p \ll n$), maximizing the data likelihood over all possible tree structures may already lead by itself to overfitting. We therefore consider a *structural* regularization of the Chow-Liu method, by modifying its optimization criterion of eqn. (4), so as to penalize model complexity in terms of its number of edges $|T|$,

$$T_{CL}^{\lambda}(\mathbf{D}) = \arg\max_{T} \sum_{(X_i,X_j)\in\mathcal{E}(T)} I_{\mathbf{D}}(X_i;X_j) - \lambda|T| \ , \tag{6}$$

where T is now allowed to be a forest (*at most* one path between any two nodes).

The optimal solution to this problem can be obtained by modifying the greedy Chow-Liu algorithm, to return the 'forest model' as soon as the next edge to be included provides an information quantity $I_{\mathbf{D}}(X_i;X_j)$ smaller than λ. Furthermore, as for supervised decision tree growing [33], we notice that penalizing in this way the tree complexity is tantamount to using a hypothesis test for checking independence of the next pair of variables to be included; such a test may be formulated by comparing the quantity $2p(\ln 2)I_{\mathbf{D}}(X_i;X_j)$ (χ-square distributed under independence, with a degree of freedom of 1 for binary variables) to a critical value depending on a postulated p-value, say $\alpha = 0.05$ or smaller. This means that an arc relating to a pair of variables (X_i, X_j) such that $2p(\ln 2)I_{\mathbf{D}}(X_i;X_j)$ computed from the dataset is smaller than the χ-square statistic threshold computed for α will never be included in the forest by our modified algorithm.

To take advantage of the first iteration of the algorithm, we use the computations performed for building a first tree of the mixture by the Chow-Liu algorithm to identify those pairs of variables whose mutual information is above the threshold, and we then consider only the set $\mathcal{S}$ of those latter pairs of variables for building trees composing the rest of the mixture (see Algorithm 5).

Algorithm 5 (Pre-pruned (bagged) Chow-Liu trees (PMBCL))

```
1. $\mathcal{S} = \{\}$, $MI = [0]_{n\times n}$
2. Repeat for $i_1, i_2 > i_1$, $i_1, i_2 \in 1,\cdots,n$
   if $I_{\mathbf{D}}(X_i;X_j) > \lambda(\alpha)$
   (a) $MI[i_1,i_2] = MI[i_2,i_1] = I_{\mathbf{D}}(X_{i_1};X_{i_2})$
   (b) $\mathcal{S} = \mathcal{S} \cup (i_1,i_2)$
3. $\hat{\mathcal{T}} = \mathbf{MWST}(MI)$
4. Repeat for $j = 2,\cdots,m$:
   (a) $\mathbf{D}'$ = bootstrap($\mathbf{D}$)
   (b) Repeat for $k = 1,\cdots,|\mathcal{S}|$:
       i. $(i_1,i_2) = GetIndices(\mathcal{S}[k])$
       ii. $MI[i_1,i_2] = MI[i_2,i_1] = I_{\mathbf{D}'}(X_{i_1};X_{i_2})$
   (c) $\hat{\mathcal{T}} = \hat{\mathcal{T}} \cup \mathbf{MWST}(MI)$
5. Return $\hat{\mathcal{T}}$, $\mu = \{1/m,\cdots,1/m\}$.
```

The complexity of Algorithm 5 is $\mathcal{O}(n^2 + mK(\alpha)\log(K(\alpha))$, i.e. similar to that of Algorithm 4 (where $K = n\ln n$); its first term is also independent of the mixture size m and its second term now depends on the effect of the chosen value of α on the number of candidate edges $K(\alpha) \equiv |\mathcal{S}|$ retained in the skeleton (the smaller α, the smaller $K(\alpha)$, in a dataset size dependent fashion).

5 Experiments

Here we empirically compare our algorithms of Sect. 4 to the baseline methods of Sect. 3. To this end, we use simulated target densities that are represented by synthetic bayesian networks over binary variables. Each structure is randomly drawn by considering variables sequentially, by uniformly drawing the number of parents for each X_i in $[0, max(5, i-1)]$ and by randomly selecting these parents in $\{X_1, ..., X_{i-1}\}$. Parameters of the networks are drawn from uniform Dirichlet distributions [30]. We present results with $n = 200$ or $n = 1000$ variables, and we performed our analysis based on data sets of $p = 200, 600, 1000$ observations, i.e. small samples given the number of variables. All results are averaged over 5 target densities and 6 learning sets for each density.

We focussed the analysis on the merit of the estimation of the probability distribution. We assessed the quality of each generated mixture by the Kullback-Leibler divergence [20], an asymmetric measure of similarity of a given density $P_{\hat{\mathcal{T}}}$ to a target density P, defined by

$$D_{KL}(P \mid\mid P_{\hat{\mathcal{T}}}) = \sum_{X \in \mathcal{X}} P(X) \log_2 \left(\frac{P(X)}{P_{\hat{\mathcal{T}}}(X)} \right) \quad . \tag{7}$$

But for computational reasons (considering all 2^n possible configurations of $\mathcal{X}$ is not feasible) this score was approximated by a Monte-Carlo procedure:

$$\hat{D}_{KL}(P \mid\mid P_{\hat{\mathcal{T}}}) = \frac{1}{N} \sum_{X \sim P}^{N} \log_2 \left(\frac{P(X)}{P_{\hat{\mathcal{T}}}(X)} \right) \quad , \tag{8}$$

where we used one test set of 50000 independent observations to estimate all the models inferred for a given target density.

For a given data set, we applied all algorithms and compared their results to the target density. Except for the Chow-Liu algorithm that produces a single tree, all mixture models are evaluated for growing numbers of terms (m=1,10,20...) up to 500 for 200 variables, in order to assess the convergence of the different methods (especially Algorithm 3), and up to 200 trees for 1000 variables to investigate the impact of the number of variables.

Parameters of all trees are learned from the full training sets (i.e. not from the bootstrap replicas that are used only to generate the structures), by maximizing the posterior likelihood of the data set based on uniform Dirichlet priors [30].

5.1 Results in Terms of Accuracies

Let us start by an evaluation of the relative accuracy performances of the different algorithms in the case of 1000 variables and 200 observations. Figure 1 displays the Kullback-Leibler divergence (vertical axis) with respect to the target density for the single CL tree (Algorithm 1) and for the other methods as a function of the mixture size m (horizontal axis). The PMBCL method (Algorithm 5) is tested here with 4 values of its parameter α: $1E^{-1}, 5E^{-2}, 5E^{-3}, 5E^{-4}$.

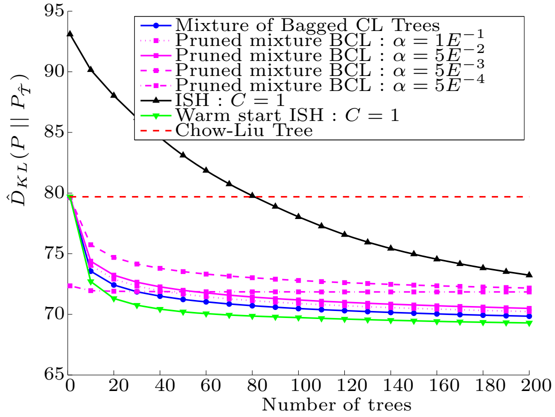


Fig. 1. A comparison between all methods presented in this paper shows the superiority of model averaging methods (with $n = 1000$ and $p = 200$). Horizontal axis: ensemble size m; vertical axis KL divergence to the target density estimated by Monte-Carlo and averaged over 5 target densities and 6 training sets.

Looking first at $m = 1$ (initial values of all curves), we observe that all but two methods start at the same point as the CL tree: ISH (Algorithm 3) is significantly worse, while the strongly regularized PMBCL tree at $\alpha = 5E^{-4}$ is significantly better[1]. For larger values of m, all the considered mixtures monotonically improve, some more quickly than others, and for sufficiently high values of m they all are quite superior to a single CL tree. For PMBCL, the smaller α, the lesser the improvement rate; actually, for $\alpha = 5E^{-4}$, its improvement rate is so small that it is quickly overtaken by the Mixture of Bagged CL Trees (at $m = 30$) and later on by PMBCL with $\alpha = 5E^{-2}$ (at $m = 60$). On the other hand, Warm Start ISH and PMBCL for α sufficiently large display comparable performances and the same convergence rate than Bagged CL Trees.

These results confirm the superiority of the model averaging approach, and they also suggest the interest of trying to limit the complexities of the individual trees in PMBCL and to correctly initialize the inertial approach, given their computational complexity advantage with respect to raw bagging (see below).

In order to allow a better understanding of the influence of α on the behavior of PMBCL, Table 1 lists the number of edges in the skeleton S and in the first tree T_1 for different values of α. It comes as no surprise that those numbers are decreasing with α. Note how the number of edges in T_1 is almost at the maximum ($n - 1 = 999$) for $\alpha \in \{1E^{-1}, 5E^{-2}, 5E^{-3}\}$, whose curves start at the same performance as the CL tree, while the smallest α ($5E^{-4}$) leads to a much smaller tree. These numbers also show that the skeleton in that last case has only a few edges more than the first tree. This is in accordance with the very small improvement in the performance of the method when the mixture

[1] Standard deviations of KL divergences, not reported for the sake of legibility, are about 20 times smaller than the average differences that we comment.

Table 1. Impact of the parameter α on the number of edges in PMBCL, averaged on 5 densities times 6 data sets for $n = 1000$ variables and $p = 200$ samples

	Numbers (% of the total) for $\alpha =$			
	$1E^{-1}$	$5E^{-2}$	$5E^{-3}$	$5E^{-4}$
Edges in T_1	998	997.9	993.2	626.8
Edges in S	52278(10.5%)	26821(5.36%)	3311(0.66%)	683 (0.13%)

is expanded and its fast convergence: the skeleton is so small that only a few different trees can be built with those edges, and the mixture quickly has them.

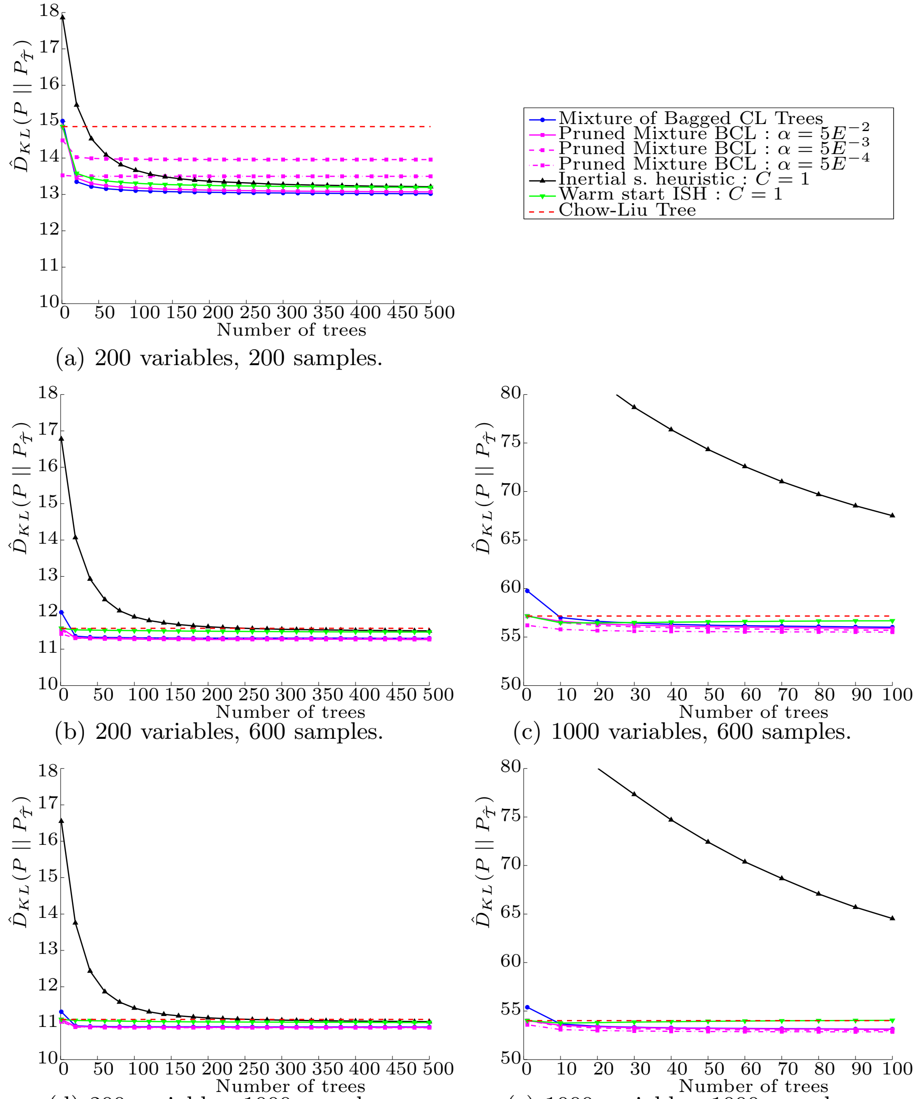
On the other hand, when the skeleton is larger, tree structures learned on bagged replica have more freedom, which allows the consideration of more candidate structures and leads to a more effective variance reduction.

Effect of the Learning Set Size *p*. To further analyze the relative behavior of the different methods, we increased the size p of learning samples to 600 and 1000. Results, reported in Figs 2(c,e), show that the most noticeable change is that the different methods now start from different initial points: the CL tree becomes initially better than a tree learned on a bagged replica. The advantage of the first PMBCL tree with the smallest value α is decreasing. We deem that both observations are a consequence of the improved precision of the mutual information estimate derived from a larger data set.

Now that the estimations of the "good" edges are better, reducing α seems to have an opposite effect on the improvement rate of PMBCL. Notice that, while at $m = 100$, the lowest α still seems better on average, confidence intervals (not displayed) suggest that the different methods cannot really be distinguished.

The Warm Start ISH is now doing far worse than the Mixture of Bagged CL Trees. We conjecture that the larger sample size leads to less variation in the mutual informations computed from bootstrap replicas, leading to slower moves in the space of tree structures for this method.

Effect of the Problem Dimensionality *n*. Modifying the number of variables has mostly an effect on the ISH methods, since it impacts the relative number of edges considered at each iteration, and thus the exploration speed of the MI matrix. Smaller numbers of variables therefore should accelerate the convergence of this method. Figures 2(a,b,d), provide a global picture of the relative performances of the considered methods, with $n = 200$ and over a longer horizon $m = 500$ of averaging. These simulations show that both inertial methods converge to the same point. Therefore, and despite a better improvement rate at the beginning, sampling structures far from the optimal one (in the original ISH method) does not improve the mixture. Based on this observation, one might actually be tempted to remove the first terms of that mixture, hoping for an improved convergence speed. We however believe that considering all edges in the first step of the method (the Warm Start variant) is more productive, since the method is directly initialized in the neighborhood of good structures.



(a) 200 variables, 200 samples.

(b) 200 variables, 600 samples.

(c) 1000 variables, 600 samples.

(d) 200 variables, 1000 samples.

(e) 1000 variables, 1000 samples.

Fig. 2. Overview of accuracy performances of the different algorithms described in this paper, with $n = 200$ or $n = 1000$ variables (left vs right), and for increasing sample sizes p (200, 600 and 1000, from top to bottom). Vertical axis: KL divergence to the target density estimated by Monte Carlo on 50,000 test observations, averaged over 5 target densities and 6 learning sets for each one. Horizontal axis: number m of mixture terms used by the different methods (except for the CL tree method, using a single tree).

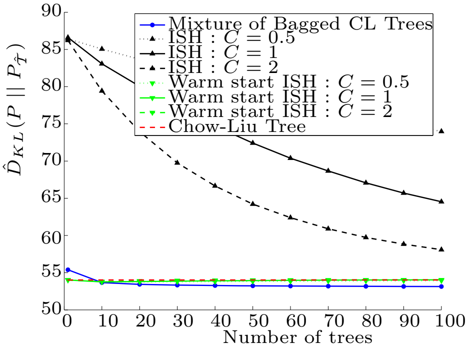


Fig. 3. Modifying the number of edges considered at each step only affects ISH when all edges are not considered in the first iteration (shown here for $n = 1000$, $p = 1000$)

Inertial Search Heuristics. Modifying the number of edges considered at each iteration in both variants of ISH, as depicted in Fig. 3, shows that the exploration of the MI matrix affects the convergence of the base method. Indeed, doubling ($C = 2$) or dividing by two ($C = 0.5$) the number of edges explored has a huge impact on its convergence, while it hardly affects the Warm Start version.

5.2 Computing Times

Our experiments were performed on a grid running ClusterVisionOS and composed of pairs of Intel L5420 2.50 Ghz processors with either 16 or 32 GB of RAM. Due to the environment, run time for a method can vary a lot, and we therefore decided to report relative minimum running time for every method. Those results are displayed in Table 2 and 3 for respectively 200 and 1000 variables / 500 and 100 trees. Results for PMBCL are reported for $\alpha = 0.005$.

Those numbers show that the proposed methods (lower part of the table) are roughly an order of magnitude faster than the standard bagging method, and this relative speed-up is stronger in the higher dimensional case. Also, as we saw from the accuracy results, these methods converge as quickly as bagging.

If one is considering parallelizing those methods at a high level, namely by computing trees individually on different cores, Bagged mixtures of CL Trees and PMBCL are the best candidates, since the trees in these methods are independent (independent conditionally on the first tree in the case of PMBCL). In the two ISH methods, each tree depends on the previous one, and parallelizing is hence more difficult. But, at a lower level, all algorithms could take advantage of the parallelization of the computation of a MWST.

Overall, the PMBCL method appears as the most appealing method; it always combines fast convergence (as fast as bagging) when the number of terms of the mixture is increased and, from the computational point of view, it is also the most efficient one among those that we investigated, about 20-30 times faster than bagging in realistic conditions; furthermore it is easy to parallelize. Neverteless, the inertial heuristic with warm start is competitive as well.

Table 2. Serial minimum computing times (given for $n = 200$ variables)

Method	Complexity	running time (500 trees - except CL)		
		200 samples	600 samples	1000 samples
Chow-Liu	$n^2 \log(n)$	1	3.07	5.3
Bagged CL Trees	$mn^2 \log(n)$	532	1531	2674
ISH	$mn \log(n)$	45	186	432
PMBCL	$n^2 + mK(\alpha)\log(K(\alpha))$	21	82	191
Warm Start ISH	$n^2 \log(n) + mn \log(n)$	45	192	406

Table 3. Serial minimum computing times (given for $n = 1000$ variables)

Method	Complexity	running time (100 trees - except CL)		
		200 samples	600 samples	1000 samples
Chow-Liu	$n^2 \log(n)$	37	98	174
Bagged CL Trees	$mn^2 \log(n)$	5037	11662	19431
ISH	$mn \log(n)$	181	800	1433
PMBCL	$n^2 + mK(\alpha)\log(K(\alpha))$	139	612	1005
Warm Start ISH	$n^2 \log(n) + mn \log(n)$	218	766	1359

Note that convergence speed may vary between methods, and some might require fewer iterations before performance (almost) stabilizes.

6 Conclusion

In this paper we have studied variance reduction oriented model averaging techniques for density estimation, using probabilistic graphical models and more precisely mixtures of Markov trees. Those models are particularly suited for problems defined on very high dimensional spaces due to their scalability.

The contributions of this paper are the proposal of algorithms for learning mixtures of Markov trees designed to approach the quality of approximation of mixtures of bagged Chow-Liu trees at a lower computational cost, and the study of their main properties. The bottleneck of the baseline bagging method is the quadratic number of edges considered for building the structure of each Markov tree of the ensemble. This is problematic since it may lead to restricting the total number of trees in the mixture, while on the other hand larger numbers of trees would yield more accurate models. The main idea behind our proposals is to use the information obtained from the computation of a first tree of the mixture so as to simplify the computation of the subsequent trees of the mixture.

We have demonstrated on synthetic datasets the interest of Markov trees averaging over regularizing a single Chow-Liu tree. For example, when enough Bagged Chow-Liu trees are averaged, they outperform that single model. Likewise, we have shown that our approximation schemes match the accuracy of bagging better than existing alternatives. Among the proposed methods, the most robust and computationally efficient one (PMBCL) defines a set of candidate edges by selecting all edges computed at the first iteration that are better than a constant complexity “edge penalty”, and subsequently only considers those edges for building remaining ensemble terms. The approximation schemes that we have proposed were in our experiments one order of magnitude faster than Mixtures of Bagged Chow-Liu Trees.

Other variants of our methods could be investigated in the future. For example, it might be interesting to perform a looser selection of edges at the first iteration, and to include additional regularization when learning subsequent terms, or vice versa. This might further ease the calibration of the tradeoff between computational complexity gains and variance reduction potential.

Acknowledgments. François Schnitzler is supported by a F.R.I.A. scholarship. Pierre Geurts is a research associate of the FNRS, Belgium. This work was also funded by the Biomagnet IUAP network of the Belgian Science Policy Office and the Pascal2 network of excellence of the EC. The scientific responsibility is the authors’.

References

1. Aliferis, C., Statnikov, A., Tsamardinos, I., Mani, S., Koutsoukos, X.: Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. JMLR 11, 171–234 (2010)
2. Ammar, S., Leray, P., Defourny, B., Wehenkel, L.: Probability density estimation by perturbing and combining tree structured Markov networks. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS, vol. 5590, pp. 156–167. Springer, Heidelberg (2009)
3. Ammar, S., Leray, P., Schnitzler, F., Wehenkel, L.: Sub-quadratic Markov tree mixture learning based on randomizations of the Chow-Liu algorithm. In: The Fifth European Workshop on Probabilistic Graphical Models, pp. 17–24 (2010)
4. Ammar, S., Leray, P., Wehenkel, L.: Sub-quadratic Markov tree mixture models for probability density estimation. In: 19th International Conference on Computational Statistics (COMP-STAT 2010), pp. 673–680 (2010)
5. Auvray, V., Wehenkel, L.: On the construction of the inclusion boundary neighbourhood for Markov equivalence classes of bayesian network structures. In: Proceedings of 18th Conference on Uncertainty in Artificial Intelligence, pp. 26–35 (2002)
6. Auvray, V., Wehenkel, L.: Learning inclusion-optimal chordal graphs. In: Proceedings of 24th Conference on Uncertainty in Artificial Intelligence, pp. 18–25 (2008)
7. Bach, F.R., Jordan, M.I.: Thin junction trees. In: Advances in Neural Information Processing Systems, vol. 14, pp. 569–576. MIT Press, Cambridge (2001)
8. Breiman, L.: Arcing classifiers. Tech. rep., Dept. of Statistics, University of California (1996)
9. Chazelle, B.: A minimum spanning tree algorithm with inverse-Ackermann type complexity. J. ACM 47(6), 1028–1047 (2000)
10. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Trans. Inf. Theory 14, 462–467 (1968)
11. Cooper, G.: The computational complexity of probabilistic inference using bayesian belief networks. Artificial Intelligence 42(2-3), 393–405 (1990)
12. Efron, B., Tibshirani, R.: An introduction to the bootstrap. Chapman & Hall, Boca Raton (1993)

13. Elidan, G.: Bagged structure learning of bayesian network. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (2011)
14. Elidan, G., Gould, S.: Learning bounded treewidth bayesian networks. JMLR 9, 2699–2731 (2008)
15. Friedman, N., Goldszmidt, M., Wyner, A.: Data analysis with bayesian networks: A bootstrap approach. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 196–205. (1999)
16. Friedman, N., Nachman, I., Peér, D.: Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 206–215 (1999)
17. Kirshner, S., Smyth, P.: Infinite mixtures of trees. In: ICML 2007: Proceedings of the 24th International Conference on Machine Learning, pp. 417–423. ACM, New York (2007)
18. Koller, D., Friedman, N.: Probabilistic Graphical Models. MIT Press, Cambridge (2009)
19. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 7(1), 48–50 (1956)
20. Kullback, S., Leibler, R.: On information and sufficiency. Ann. Math. Stat. 22(1), 79–86 (1951)
21. Kumar, M.P., Koller, D.: Learning a small mixture of trees. In: Advances in Neural Information Processing Systems, vol. 22, pp. 1051–1059 (2009)
22. Kwisthout, J.H., Bodlaender, H.L., van der Gaag, L.: The necessity of bounded treewidth for efficient inference in bayesian networks. In: the 19th European Conference on Artificial Intelligence, pp. 623–626 (2010)
23. Liu, H., Xu, M., Haijie Gu, A.G., Lafferty, J., Wasserman, L.: Forest density estimation. JMLR 12, 907–951 (2011)
24. Madigan, D., Raftery, A., Wermuth, N., York, J., Zucchini, W.: Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam's Window. Journal of the American Statistical Association 89, 1535–1546 (1994)
25. Meila, M., Jordan, M.: Learning with mixtures of trees. JMLR 1, 1–48 (2001)
26. Meila, M., Jaakkola, T.: Tractable bayesian learning of tree belief networks. In: Proc. of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp. 380–388. Morgan Kaufmann, San Francisco (2000)
27. Ormoneit, D., Tresp, V.: Improved gaussian mixture density estimates using bayesian penalty terms and network averaging. In: Advances in Neural Information Processing Systems, pp. 542–548. MIT Press, Cambridge (1995)
28. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, San Francisco (1988)
29. Robinson, R.W.: Counting unlabeled acyclic digraphs, vol. 622. Springer, Heidelberg (1977)
30. Schnitzler, F., Leray, P., Wehenkel, L.: Towards sub-quadratic learning of probability density models in the form of mixtures of trees. In: 18th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2010), Bruges, Belgium, pp. 219–224 (2010)
31. Shahaf, D., Chechetka, A., Guestrin, C.: Learning thin junction trees via graph cuts. In: Artificial Intelligence and Statistics (AISTATS), pp. 113–120 (2009)
32. Tarjan, R.E.: Data structures and network algorithms. Society for Industrial and Applied Mathematics, Philadelphia (1983)
33. Wehenkel, L.: Decision tree pruning using an additive information quality measure. In: Uncertainty in Intelligent Systems, pp. 397–411 (1993)

Resource-Aware On-line RFID Localization Using Proximity Data

Christoph Scholz[1], Stephan Doerfel[1],
Martin Atzmueller[1], Andreas Hotho[2], and Gerd Stumme[1]

[1] Knowledge & Data Engineering Group, University of Kassel,
34121 Kassel, Germany
{scholz,doerfel,atzmueller,stumme}@cs.uni-kassel.de

[2] Data Mining and Information Retrieval Group, University of Würzburg,
D-97074 Würzburg, Germany
hotho@informatik.uni-wuerzburg.de

Abstract. This paper focuses on resource-aware and cost-effective indoor-localization at room-level using RFID technology. In addition to the tracking information of people wearing active RFID tags, we also include information about their proximity contacts. We present an evaluation using real-world data collected during a conference: We complement state-of-the-art machine learning approaches with strategies utilizing the proximity data in order to improve a core localization technique further.

1 Introduction

While approaches for outdoor localization can utilize various existing global sources, e.g., GPS signals, mobile broadcasting signals, or wireless network signatures, methods for indoor localization usually apply special installations (e.g., RFID or Bluetooth readers), or require extensive training and calibration efforts.

In this paper, we propose an approach for indoor localization using active RFID technology: We focus on a cost-effective and resource-aware solution that requires only a small number of RFID readers (Figure 1). Furthermore, our method can be applied to installations, where readers cannot be positioned freely. The latter constraint is encountered often, especially in buildings under monumental protection. Our application context is a conference, where conference participants wear active RFID tags for tracking, for memorizing their contact information, and for the personalization of conference services. Therefore, we present an analysis of data collected in a real-life context, in contrast to scenarios that examine RFID localization in laboratory experiments, e.g., [18][12]. In Section 3.1 we discuss additional challenges, that such an application faces and that are difficult to implement in simulated scenarios. We consider a real-life localization problem at room-level, i. e., the task to determine the room, that a person is in at a given point in time.

Our contribution is three-fold: We present an analysis of the contact and proximity data in order to prove the validity and applicability for the sketched application. Additionally, we evaluate the benefits of several state-of-the-art machine

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 129–144, 2011.

learning techniques for predicting the locations of participants at the room-level. We propose to utilize the (proximity) contacts of participants for improving the predictions of a given core localization algorithm. We evaluate the impact of different strategies considering the top performing machine learning algorithm. The real-world evaluation data was collected at the LWA 2010 conference (of the German Association of Computer Science) in Kassel, Germany[1].

The rest of the paper is structured as follows: Section 2 discusses related work. After that, Section 3 describes the approach for resource-aware localization at room-level using RFID technology. Next, Section 4 features the evaluation of the approach utilizing several machine learning algorithms and different strategies for implementing the proximity contacts. Finally, Section 5 concludes the paper with a summary and interesting directions for future research.

2 Related Work

The Global Positioning System (GPS) is the most widely used localization system for outdoor positioning. It is based on a network of 24-30 satellites placed in the orbit. One of the drawbacks of GPS is that it cannot be used for indoor localization, because its signals are blocked by most construction materials. Therefore, the research on indoor localization systems has received great interest during the last decade.

For indoor localization several algorithms have been proposed, usually based on angle of arrival (AoA) [19], time of arrival (ToA) [15] or time difference of arrival (TDoA) [20] methodologies. On the one hand, these methods are highly accurate in estimating the position of an object; on the other hand they consume a lot of energy. Furthermore, they require expensive hardware and an extensive deployment of suitable infrastructure. Another class of localization algorithms estimates the position of a target based on the received signal strength [12]. Most of these approaches use the log-distance path loss model [21] to estimate the distance from the object to at least three reference points. Then, the possible position of the object is calculated using triangulation. The disadvantage of this approach is that propagation effects such as reflection, multi-path-fading or phase-fluctuations limit the precision of positioning.

Scene analysis is another option to estimate the position of an object [4][18][6]. Usually, this technique works in two phases, the off-line learning phase and the online localization phase. In the off-line phase, data about the received signal strengths (RSS) for each point in the localization area is stored in a database to save the localization points. In the online phase two scene analysis techniques of predicting the position exist: k-nearest-neighbor (KNN) and probabilistic methods. KNN predicts the position by finding the k closest fingerprints in the database. The estimated location is the (weighted) centroid of the corresponding k locations. The probabilistic model selects the location with the highest probability.

[1] http://www.kde.cs.uni-kassel.de/conf/lwa10/

In our experiments, we consider a different approach using a new generation of cost-effective and resource-aware RFID tags, i.e., tags with a low power consumption. These RFID tags (proximity tags) are developed by the SocioPatterns project[2] and the company Bitmanufaktur[3]; at the time of writing the project will soon become open-source, see the SocioPatterns web site for more information. The technical innovation of the applied tags is their ability to detect the proximity of other tags within a range of up to 1.5 meters. Due to the fact, that the human body blocks RFID signals, face-to-face contacts can then be detected. In this context, one of the first experiments using RFID tags for tracking the position of persons on room basis was conducted by Meriac et al. (cf., [16]) in the Jewish Museum Berlin in 2007. Cattuto et al. [8] added proximity sensing in the SocioPatterns project. Barrat et al. [13] did further experiments.

For several research questions, e.g., for social network analysis, it is rather interesting to combine the movement and contact data of persons. To conduct such analysis we apply active RFID tags that provide data from which we can extract positioning data as well as contact data. The proximity-tags are primarily developed for recognizing face-to-face contacts.

In the context of the presented approach, one additional problem concerns the exact positioning information: Our hardware setting does not provide information (like ToA, TDoA, AoA, RSSI, ...) used for positioning in traditional localization algorithms. Like the work of [16] we use the number of packages each RFID reader received from each RFID tag in a specific time interval (for each signal strength) to determine the users position. Compared to the work of [16] we use a fingerprint technique to estimate the location of the user. In [16] the participant is allocated to the room whose RFID readers received most packages with the weakest signal strength. This approach works fine, but it is based on the fact that at least two readers are placed in each room. Unfortunately, often it is not possible to place the RFID readers at arbitrary positions, e.g., in older buildings, or buildings with monumental protection.

Localization with proximity tags and readers as applied in our hardware setting is challenging for different reasons: First, the number of packages per second sent by the RFID tags is very low. Second, the position of RFID readers can not be chosen freely and the number of readers should be as small as possible. Third, as discussed above, the RFID readers do not offer additional information (like ToA, RSS,...) about the received packages from proximity tags. Fourth, as already described in previous work RFID properties like reflection and multipath-fading complicate the task of localization. For further reading about RFID we refer to [10,11].

In this paper, we propose a resource-aware approach for indoor localization using proximity tags. To the best of the authors' knowledge, this is the first time that the accuracy of such a localization approach is investigated in a real world application. In contrast, to the presented approach all existing literature studied their approaches under nearly optimal (laboratory) conditions, e.g., [18][12].

[2] http://www.sociopatterns.org

[3] http://www.bitmanufaktur.de

3 Resource-Aware RFID Room-Level-Localization

In the following section, we first outline the resource-aware application scenario using active RFID tags. After that, we describe the application of machine learning for room-level prediction of the tags' location. Next, we summarize the strategies for improving the accuracy of the applied methods by utilizing the proximity contacts between the applied RFID tags.

3.1 Resource-Aware RFID Application Scenario

In this paper, we aim at a flexible and resource aware approach for localization using RFID: It should require only a small number of readers, and should further allow the free placement of readers not constrained to single rooms. E. g., a reader might be assigned to several areas, or to larger areas in general. In our experience, such a setting is highly relevant for practical applications, and also needed to be taken into account for our real-world evaluation scenario. Further issues, that have to be overcome in a real-world setting are the interference between tags – if many tags are put into one location and signals transcending room boundaries, i. e., walls or ceilings.

In summary, in a real-life setup the localization problem is much more complicated compared to a simulated environment, using very many readers and resources e. g., [16] . Below, we describe the hardware and system architecture used in our localization experiment.

Hardware. For our localization experiment at the poster session of a conference we asked each participant to carry an active RFID tag (see Figure 1). The tags provide localization and proximity detection in a resource-aware and cost-effective way, which conforms to our requirements. Every two seconds each RFID tag sends one package in four different signal strengths (-18dbm, -12dbm, -6dbm, -0dbm) to RFID readers placed at fixed positions in the conference area (see Figure 2). Dependent on the signal strength the range of one package inside a building is up to 25 meters. Each package is 128 bits long, encrypted, and contains information about the tag id from the reporting RFID tag, signal strength and CRC checksum. For more details, we refer to Barrat et al. [5] and [2]. The continuous sending of RFID packages in uniform time-intervals (two seconds) gives us the opportunity of determining the package-loss of an RFID tag at each RFID reader. We use this information to create the characteristic RFID vectors. Here, we note that we do not use the package loss explicitly. Instead we use the number of packages an RFID reader receives from a tag.

One decisive factor, that makes proximity tags interesting for conference scenarios is the possibility to detect other proximity tags within a range of up to 1.5 meters. Since the human body blocks RFID signals, one can detect and analyze face-to-face contacts in this way [5]. In this work, we show that this proximity information helps to improve the localization accuracy. The information about contacts is transmitted in the fourth and strongest signal strength of the tags. Thus, a tag sometimes sends more than one package (every two seconds) in that strength, because more packages are needed to transport the contact information. Since it is not possible to store information on the tags permanently, every time-dependent information is lost, when a tag is out of the range of all RFID readers.

The RFID readers (see Figure 1) receive RFID signals and forward them to a central server via UDP where the signals are decrypted, analyzed and stored in a database. Because of resource-awareness reasons the RFID readers do not provide information like AoA or RSS of the received packages, which could help to additionally improve the accuracy of the localization results.

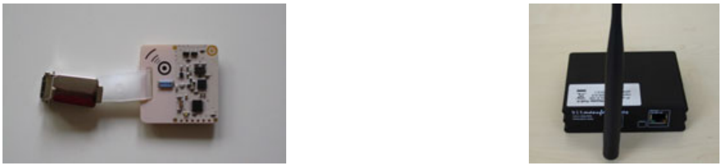


Fig. 1. Proximity tag (left) and RFID reader (right)

3.2 Machine Learning for Prediction Using RFID Data

As described in Section 3.1, each RFID tag sends one package in four different signal strengths every two seconds. Similar to most fingerprint approaches we assume that the number of packages an RFID reader received is significantly dependent on the position of the sending RFID tag, i. e., when a tag is moved away from the reader, the number of received packages will decrease. Therefore, we can determine sets of characteristic vectors (fingerprints) for each room in the conference area.

Observation Vector Space. In a setting with R RFID readers and P proximity tags, each transmitting on S different signal strengths, let l denote the length of a time window and t a point in time. Further, let $V_r^l(p,t) \in \mathbb{N}^s$ ($1 \leq r \leq R$, $1 \leq p \leq P$) be an S-dimensional vector where the s-th entry is the number of packages that RFID reader r received from proximity tag p with signal strength s in the time interval $[t-l,t]$. The vector

$$V^l(p,t) = \left(V_1^l(p,t), V_2^l(p,t), \cdots, V_R^l(p,t)\right) \qquad (1)$$

– i. e., the concatenation of the vectors $V_r^l(p,t)$ over all readers r, – is called the *package count vector* or *characteristic vector* of the proximity tag p at time t. The dimension of vector $V^l(p,t)$ is $S \cdot R$. With the parameter l one can control the influence of older signals. For longer time intervals, the probability rises that packages sent from a previous location influence the vector at the current time point t.

We consider the localization problem as a classification task. In the learning phase, we create a set of fingerprints (training data) for each room, and learn a classification model based on these fingerprints. In the online classification (localization phase) we determine the position of a participant from his current fingerprint, using the classification model. In this paper, we modify four state of the art machine learning methods for that classification task by including proximity contact information and analyze the resulting increase in their accuracy due to these contacts.

Basic Room Prediction. In the following, we outline the basic machine learning methods that we applied as a benchmark for predicting locations of the participants, and as initial methods to be complemented with the contact strategies described below. We briefly summarize their basic features, and discuss their application using the RFID data. We refer to the basic localization methods as the Loc-Basic approach.

Naive Bayes (nBay). While naive Bayes [17] is a rather simple approach, studies comparing classification algorithms have shown that the naive Bayes classifier is often comparable in performance with decision trees, while achieving high accuracy and speed being applied to large databases. Therefore, for the localization naive Bayes is a good candidate due to its learning performance and accuracy.

K-Nearest Neighbor (kNN). As a lazy learner, the k-nearest neighbor algorithm [17] is easy to setup and implement, since only a certain set of training data needs to be stored, and a suitable distance (similarity) metric be applied for retrieving a similar case for a given query. Therefore, a scenario that does not allow for long training periods favors a nearest neighbor classifier. The parameter k controls the number of neighbors considered for each prediction.

Support Vector Machines (SVM). Support vector machines [9] have become one of the benchmark techniques for machine learning approaches due to their good classification performance for a broad range of applications. Therefore, we also consider support vector machines as our basic learning strategy and benchmark method. In this scientific work we use the SVM$^{\text{light}}$ C-implementation [3] from Thorsten Joachims. For our experiments described in Section 4 we chose an RBF kernel

$$K_{a,b} = \exp\left(-\gamma||x_a - x_b||^2\right), \tag{2}$$

where x_a and x_b are package count vectors. In Section 4 we analyze the best parameter combination for parameter $\gamma \in \mathbb{R}$ and parameter $j \in \mathbb{R}$. Here, the parameter j is the cost factor, by which training errors on positive examples dominate errors on negative examples[4]. For all other parameters we chose the default values as described in [14].

Random Forest (RF). The random forest classifier [7] is an ensemble classification method: It applies a set of unpruned decision trees for classification. It can usually be learned in a cost-effective manner. Therefore, we also selected a random forest method for our set of base learners for the localization approach. In Section 4 we analyze the accuracy for different combinations of the two input parameters *mtry* (denoting the number of predictors sampled for splitting at each node) and *ntree* (the number of trees). For our experiments we use the *R*-implementation of Random Forest [1].

[4] http://svmlight.joachims.org/

3.3 Advanced Room Prediction Using Contacts

In this section we describe three simple but effective techniques which include contact information for improving the accuracy of the Loc-Basic algorithms. Let $C^w(p,t)$ denote the set of users (proximity tags) that were in contact with user p within a time interval $[t-w,t]$. Hereby, the length w of that interval is independent from the length l of the time interval used in the construction of the characteristic vectors. Assume, that we want to predict the position of user p at time t.

Mean-Approach. As input for the Loc-Basic algorithm the following vector is used:

$$V^l_{mean}(p,t) = \frac{V^l(p,t) + \sum_{q \in C^w(p,t)} V^l(q,t)}{1 + |C^w(p,t)|} . \tag{3}$$

Thus, the new characteristic vector $V^l_{mean}(p,t)$ of user p is the average over all package count vectors of the contacts of user p and of user p himself.

Max-Approach. Let $(v_p^1, \cdots, v_p^{SR})$ be the component representation of $V^l(p,t)$. As input for the Loc-Basic algorithm the vector

$$V^l_{max}(p,t) = \left(\max_{q \in C^w(p,t) \cup \{p\}} \left\{ v_q^1 \right\}, \cdots, \max_{q \in C^w(p,t) \cup \{p\}} \left\{ v_q^{S \cdot R} \right\} \right) \tag{4}$$

is used. $V^l_{max}(p,t)$ is the component-wise maximum of the characteristic vectors of user p and his contacts.

Vote-Approach. This approach consists of two phases. At first (preliminary) positions for user p and all his contact users are predicted using Loc-Basic. Then, the final prediction of p's position is established by a majority vote among all these Loc-Basic predictions.

4 Evaluation

Below, we first discuss the applied data before we describe the evaluation setting in detail. After that, we present the results of our experiments, and conclude with a comprehensive discussion.

4.1 Datasets

We utilized real-world data collected at the LWA 2010 conference in Kassel, covering the locations of tracked participants and contacts between these. In order to obtain a diverse and interesting set of observations we focused on the two hour poster session, since during that time many participants had gathered in 5 adjacent rooms. This provides us a challenging scenario for our methods. To ensure that each point in the conference area was covered, we placed 6 RFID readers at adequate positions in the conference area (see Figure 2).

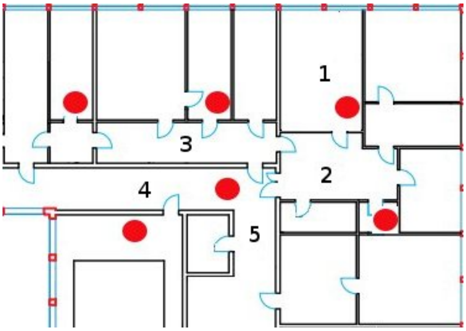


Fig. 2. Conference Area: the numbered rooms were used by participants during the poster-session, the circles mark the positions of RFID readers

We consider two kinds of tags: *user tags* and *object tags*: A user tag is a proximity tag worn by a participant during the conference. With an object tag we denote a proximity tag fixed to an unmovable object. In total, we fixed 46 object tags to several posters, tables and seats. Depending on its size we put between two and thirteen object tags in each room. The *training data* contains the first 1500 characteristic vectors collected with the object tags for each room of the conference area. Obtaining the training set took about 25 minutes.

Ground truth: In summary, 46 people took part in our localization experiment during the poster session. We collected their tag data over a duration of two hours. To evaluate the accuracy of our predictions we needed to determine the positions of the participants, for which we applied the object tags. Since the tags detect other proximity tags only within a range of up to 1.5 meters, whenever a contact between the tag of a participant tag and an object tag was recorded, we could infer that this participant was in the same room as the object tag (ground truth). In the experiments, we predicted the rooms for those vectors where the precise location could be verified with the ground truth data.

4.2 Setting

In all experiments, the target is to maximize the overall localization accuracy. The setup of the experiments contains a variety of parameters such as tuning parameters of the algorithms, parameters that control the vector space of our observations or parameters to control the set of contacts for each user at a specific time. Several of these parameters are data set dependent. Due to the nature of our setting as a social get-together most users did not switch locations very often. It is therefore possible and useful to choose large intervals to construct the observation vector space, in our experiments we chose $l = 10, 30$ and 50 seconds. However, other contexts might demand more frequent changes of the locations. In such cases fingerprints should be collected only over rather short time intervals.

Since the fourth and strongest signal strength of the tags transmits in irregular intervals (in contrast to the other signal strengths), we considered vectors including or excluding the fourth signal strength. All in all, we obtained six different datasets, in the following referred to as F_{10-3} through F_{50-4} where e. g., the vectors of F_{50-4} are collected over $l = 50$ seconds and constructed with all four signal strengths of each tag. Depending on the length of the time window l and the number of used signals, the size of training data is shown in Table 1. To include the contact information we used the mean, max and vote approach. The first parameter to choose is the length w of the time window over which we collect contacts. We experimented with five time windows: $w \in \{2, 5, 10, 20, 30\}$ (in seconds). A second parameter d is the *degree of transitive closure*, that is added to the contact set. Contact information for one user at a specific time can be sparse. In such cases it may be of help to "add more contacts" based on the rationale, that contacts between users u_1 and u_2 and between u_2 and u_3 might indicate a contact between users u_1 and u_3. This procedure of adding such (transitive) contacts can be iterated and d is the count of these iterations. Since for $d = 7$ no new contacts were produced we investigated the values $d = 0, 1, \ldots, 6$.

Table 1. Size of the ground truth dataset for different time window lengths l and numbers of signals.

F_{10-3}	F_{10-4}	F_{30-3}	F_{30-4}	F_{50-3}	F_{50-4}
135208	137126	137454	137579	137570	137586

To prevent combinatorial explosion, we structured our experiments into two parts, described below: In the first part, we applied each of our four Loc-Basic algorithms with different parameter settings to each of the six datasets. In the second part we additionally considered contact data to increase the localization quality for those parameter settings, that performed best in part one. Additionally, we conducted several experiments exploring variations of the size of the training set.

4.3 Results and Discussion - Part 1: Machine Learning Baseline

Table 2 presents the results of the first phase showing the best parameter combinations for each dataset and algorithm together with the achieved overall accuracies. We ran kNN with values for k from 5 through 200 in steps of 5. For RF we tried all combinations of $mtry = 1, \ldots, 20$ and forest sizes $ntree$ of 25 through 500 in steps of 25. SVM was run with combinations of $j = 1, \ldots, 20$ and $\gamma \in \{2, 0, -2, -4, \ldots, -18, -20\}$. Finally, the nBay does not depend on a parameter. An immediate observation is, that nBay was always outperformed by any of the other algorithms. This is not surprising as the basic assumption of nBay is the complete independence of the entries in each observation. Such independence can not be claimed for our datasets. If a reader receives, e. g., packages from a tag in its lowest signal strength, then it is much more likely that the reader will also receive packages in a higher strength from that tag. However, since we are interested in observing the boost that contact information can have on the results of a given classifier, we experimented with nBay rather than with more complex Bayes approaches taking dependencies into account.

Table 2. For each algorithm and dataset the best parameters settings and the resulting total accuracy in %. * The same accuracy was achieved with $\gamma = -12$.

base		F_{10-3}	F_{10-4}	F_{30-3}	F_{30-4}	F_{50-3}	F_{50-4}
kNN	k	50	165	125	185	180	200
	ACC	71.96	73.58	74.36	79.33	73.26	79.80
RF	$mtry$	1	1	1	2	1	4
	$ntree$	475	375	400	350	275	200
	ACC	77.44	78.03	83.66	84.53	84.18	84.78
SVM	j	1	1	7	1	13	1
	γ	-14	-14	-10^*	-18	-10	-20
	ACC	78.05	77.95	82.55	84.15	82.53	84.84
nBay	ACC	33.42	38.97	51.14	56.96	56.57	61.97

The results of the other three algorithms are between 71.96% and 84.78%. Taking into account the room layout and the hardware constraints due to our resource-aware approach, these results can be considered acceptable. As can be expected, in all cases the more sophisticated algorithms RF and SVM had higher scores than the simple kNN. Including the fourth signal strength into the datasets yielded better results than ignoring it – with one exception (SVM, F_{10-4}) where the two results differ, however only by 0.1%. Furthermore, the datasets where the package vectors are collected over 30 or 50 seconds yield better scores compared to the ones where only 10 seconds are considered.

A closer look at the influence of the algorithms parameters is presented (exemplary) in the diagrams of Figure 3. For higher values of k the accuracy of kNN rises, up to a certain level. After a (dataset dependent) threshold the accuracy almost stabilizes at that level. While the choice of the forest size *ntree* for RF did not influence the result much, the choice of the *mtry* parameter is of importance. In general, with lower values (1 through 4) the results were significantly better than for other choices. The curves of SVM fluctuate strongly on the datasets F_{10-3} and F_{10-4} and yield more stable curves for the others. In general, better results where achieved using very small values for the parameter j – in the cases where the best score was obtained with $j = 7$ or $j = 13$, the scores using $j = 1$ were not significantly lower. In all cases, results were better using lower values for γ such as -20.

4.4 Results and Discussion - Part 2: Utilizing Contact Information

In the second phase of the experiments, we employed the best parameters determined in phase one (Table 2) and included contact information to boost the localization accuracy. Tables 3, 4 and 5 present for each dataset and algorithm the best choice of the two parameters w and d and the achieved accuracy. Furthermore, for each method the lowest accuracy that was achieved with any combination of the two parameters is given. In the tables, bold numbers mark the accuracies of those methods, that performed best for the given algorithm and dataset. Italic numbers indicate accuracies, that are below the according baseline of phase one.

A first encouraging observation is, that in all experiments with the mean or max approach, the methods had a strictly positive influence on the accuracy.

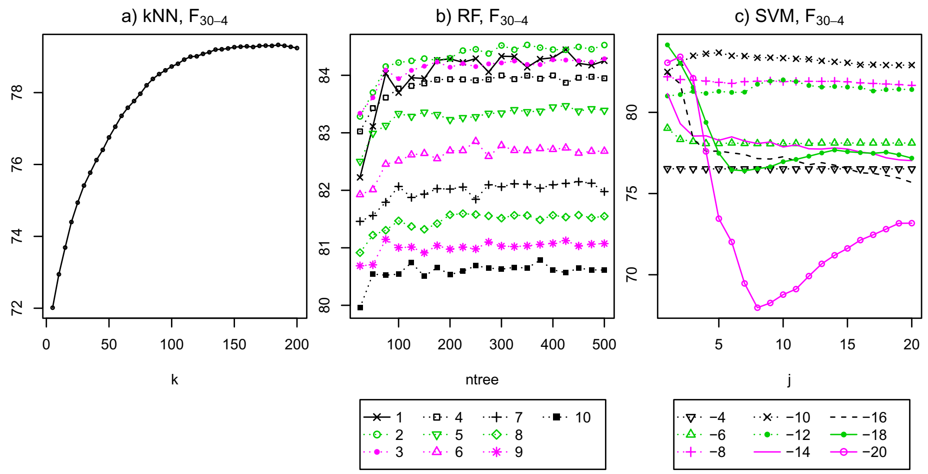


Fig. 3. Exemplary for F_{30-4}, the diagrams, showing the accuracies in % a) vs. k (kNN), b) vs. $ntree$ for different values of $mtry$ (RF) and c) vs. j for different values of γ (SVM). In c) the graphs for $\gamma = 2$, $\gamma = 0$ and $\gamma = -2$ were left out for the sake of legibility. All three are constant with accuracies 14.81%, 15.94% and 48.74%.

Table 3. For each algorithm with max aggregation and each dataset the best choices of w and d with the according accuracy in % and the minimum accuracy achieved with max aggregation.

max		F_{10-3}	F_{10-4}	F_{30-3}	F_{30-4}	F_{50-3}	F_{50-4}
kNN	w	30	20	30	30	30	30
	d	4	4	1	2	1	2
	top ACC	**80.26**	**80.28**	**83.39**	85.40	**82.78**	85.53
	min ACC	78.01	78.75	80.99	84.04	80.25	84.1
RF	w	20	20	20	20	20	5
	d	3	2	6	1	6	1
	top ACC	**84.94**	**85.49**	**89.59**	**89.96**	**88.94**	87.53
	min ACC	83.25	83.99	88.31	88.92	87.95	86.73
SVM	w	20	20	20	20	20	20
	d	2	2	5	1	2	1
	top ACC	**84.43**	**85.46**	88.16	89.14	88.65	88.72
	min ACC	82.89	83.73	87.09	88.33	87.42	88.06
nBay	w	30	30	30	30	30	30
	d	3	4	2	6	1	1
	top ACC	**50.60**	**56.81**	**65.80**	**71.95**	69.55	**76.57**
	min ACC	44.40	50.13	61.30	66.70	66.77	72.50

Only voting performed in some cases worse than the baseline, mainly for nBay. For nBay we attribute this to the fact, that the voting scheme is a probabilistic method. Since nBay itself has only a very low accuracy, it is likely that among the votes many are in fact false predictions. Thus, the probability of a wrong classification even rises.

With one exception the best results were always achieved using max or mean aggregation. Here, including the contact information yielded significant boosts

Table 4. For each algorithm with mean aggregation and each dataset the best choices of w and d with the according accuracy in % and the minimum accuracy achieved with mean aggregation.

mean		F_{10-3}	F_{10-4}	F_{30-3}	F_{30-4}	F_{50-3}	F_{50-4}
kNN	w	10	20	30	20	30	30
	d	1	1	1	2	2	2
	top ACC	75.79	79.55	80.68	**85.62**	79.63	**86.24**
	min ACC	75.19	78.52	79.35	84.30	78.24	84.73
RF	w	10	10	20	20	30	30
	d	2	3	6	3	4	5
	top ACC	79.51	80.65	88.04	88.33	88.29	88.83
	min ACC	78.00	79.31	86.57	87.29	86.83	87.57
SVM	w	20	20	30	30	30	20
	d	2	3	2	2	2	2
	top ACC	83.96	85.01	**88.43**	**89.49**	**88.89**	**89.39**
	min ACC	82.56	83.39	86.91	88.33	87.26	88.32
nBay	w	30	30	30	30	30	30
	d	5	5	2	2	2	2
	top ACC	49.61	52.62	65.22	68.88	**71.26**	75.54
	min ACC	43.93	48.20	60.23	64.31	66.26	70.59

Table 5. For each algorithm with voting aggregation and each dataset the best choices of w and d with the according accuracy in % and the minimum accuracy achieved with voting aggregation. * The same accuracy was achieved with $d = 4$

vote		F_{10-3}	F_{10-4}	F_{30-3}	F_{30-4}	F_{50-3}	F_{50-4}
kNN	w	20	30	20	20	20	20
	d	2	3	2*	3	3	3
	top ACC	77.39	79.71	80.02	85.17	78.63	85.94
	min ACC	76.11	78.35	78.82	83.80	77.69	84.49
RF	w	10	10	20	10	20	20
	d	5	1	1	2	2	3
	top ACC	81.32	81.67	86.77	87.55	87.81	**88.97**
	min ACC	80.75	81.14	86.27	87.08	87.03	87.88
SVM	w	20	20	5	20	2	30
	d	4	4	0	2	0	2
	top ACC	81.35	81.09	83.22	86.75	82.76	87.46
	min ACC	80.05	79.51	*81.29*	85.55	*80.86*	86.17
nBay	w	2	2	2	2	2	2
	d	0	0	0	0	0	0
	top ACC	*33.05*	*38.70*	*50.90*	*56.53*	*56.10*	*61.31*
	min ACC	*31.48*	*37.54*	*50.39*	*55.81*	*55.22*	*60.66*

Table 6. For each algorithm (in its best performing aggregation parametrization) the fraction of data for which contact information is available (in %) and a comparison of prediction accuracy of the algorithms without contacts and those using the best performing method of contact data aggregation.

	kNN	RF	SVM	nBay
strategy	F_{50-3}	F_{10-3}	F_{10-4}	F_{10-4}
contact fraction	69.23	65.01	64.91	69.3
contact base ACC	74.33	76.83	77.78	33.05
contact best ACC	88.09	88.18	89.34	58.80
boost	13.76	11.34	11.57	25.74

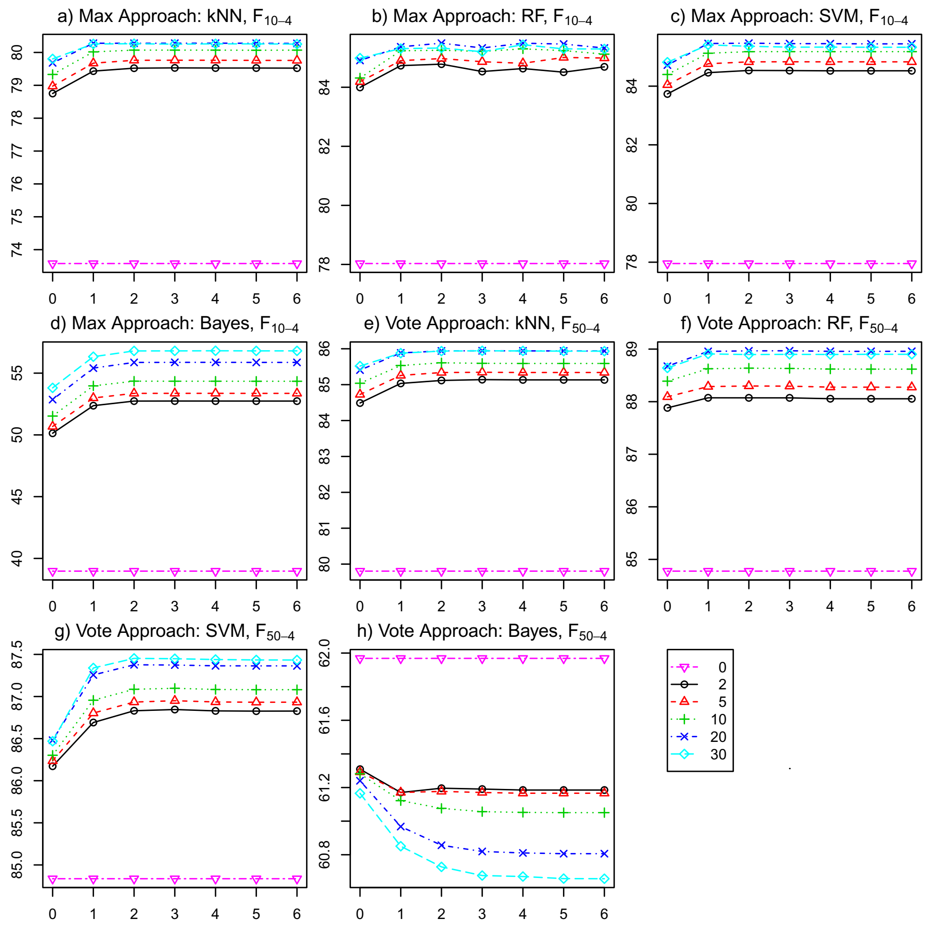


Fig. 4. a) through d) present the accuracies in % of all experiments with the max approach on F_{10-4}, e) through h) those of the experiments with the vote approach on F_{50-4}. For several choices of w, the accuracy is plotted vs. the degree of transitivity d.

in overall accuracy: up to an additional 9.52% for kNN (F_{50-3}), 7.5% for RF (F_{10-3}), 7.51% for SVM (F_{10-4}) and 17.84% for nBay (F_{10-4}). These results are clear evidence, that the contact information can support the localization approach significantly. Even stronger evidence for that presents Table 6. This table shows for the above mentioned four settings the fraction of test data where contact information (depending on the parameters d and w, chosen as in Table 3) is available (contact fraction). Further, given are the prediction accuracies on only that fraction of the dataset of both, the Loc-Basic algorithms (contact base acc) and the best contact boosted algorithms (contact best acc). Boost denotes the additional gain of accuracy due to the inclusion of the contact data. Here, the scores of kNN, RF and SVM profit with more than 11% while the

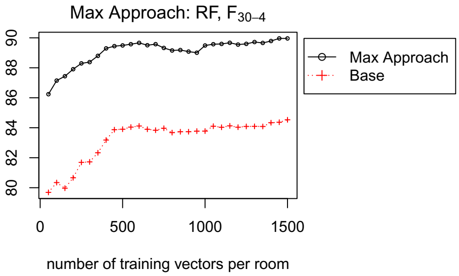


Fig. 5. The accuracy in % vs. the number of training samples per room

accuracy of nBay increases by more than 25%. Our best performing algorithm with respect to the complete test set (RF with max aggregation using F_{30-4}) yields a prediction accuracy of 92.69% if applied to that part dataset for that contact information is available.

Next, we investigated the influence of the parameters d and w. As can be seen in the Tables 3, 4, 5 the fluctuation of the accuracy for different parameter combinations was rather low, often less than 1%. Figure 4 displays exemplary for each algorithm the results of the max approach for the F_{10-4} dataset and of the vote approach for the F_{50-4} dataset. The behavior of the accuracy using the mean approach was generally similar to that of the max approach. The parameter d usually had only a small influence. In most experiments only the difference between $d = 0$ and $d = 1$ was significant. For the values $1, \ldots, 6$ the accuracy stayed almost constant. Variations of the w parameter also caused similar behavior throughout the experiments. In those, where the contacts had positive influence on the accuracy, the choices $w = 20$ or $w = 30$ delivered the best results, while $w = 10$ usually was better than $w = 5$ or $w = 2$.

Furthermore, we analyzed the influence of the training set size. We applied the method from our previous experiments that performed best (RF with $ntree =$ 350 , $mtry = 2$, $w = 20$ and $d = 1$ using the max approach on F_{30-4}) to classify with models based on differently sized training sets. Figure 5 shows the resulting accuracies compared to those of the according Loc-Basic method. Up to 450 samples per room, increasing the training size increases the accuracy. Afterwards the accuracy increases only little or decreases in some cases. The distance between the curves (the boost due to the contacts) is almost constant, only for very small training set sizes it is slightly larger.

5 Conclusions

In this paper, we have presented an approach for cost-effective and resource-aware localization at room level using RFID-tags. We evaluated several state-of-the-art machine-learning algorithms in this context, complemented by novel techniques for improving these using the RFID (proximity) contacts. The results of the experiments yielded several reasonable values for the applicable parameters. For the simpler algorithms, they could also have been learned in a short preceding training phase, which demonstrates the broad applicability of the approach in the sketched resource-aware setting.

In the presented experiments we always considered training data collected by the object tags. In future work, we aim to analyze the accuracy of the proposed approach using the user tags in more detail. An extended analysis concerns using all available and also no contact information at all, respectively, when we consider the user tags for obtaining the training data. Furthermore, we plan to focus on optimizing the applied parameter combinations, i.e., number of readers, number of packages per second, etc., in order to increase the accuracy further. Testing our algorithms in WiFi and GPS based localization settings is also another interesting option for future work.

Acknowledgements. This work has been supported by the VENUS research cluster at the interdisciplinary Research Center for Information System Design (ITeG) at Kassel University. We utilized active RFID technology which was developed within the SocioPatterns project, whose generous support we kindly acknowledge. We also wish to thank Milosch Meriac from Bitmanufaktur in Berlin for helpful discussions regarding the RFID localization algorithm. Our particular thanks go the SocioPatterns team, especially to Ciro Cattuto, who enabled access to the Sociopatterns technology, and who supported us with valuable information concerning the setup of the RFID technology.

References

1. CRAN - Package randomForest, http://cran.r-project.org/web/packages/randomForest/index.html
2. OpenBeacon Active RFID Project, http://www.openbeacon.org
3. SVM-Light Support Vector Machine, http://svmlight.joachims.org/
4. Bahl, P., Padmanabhan, V.N.: RADAR: An In-Building RF-Based User Location and Tracking System. In: INFOCOM, pp. 775–784 (2000)
5. Barrat, A., Cattuto, C., Colizza, V., Pinton, J.F., den Broeck, W.V., Vespignani, A.: High Resolution Dynamical Mapping of Social Interactions with Active RFID. CoRR abs/0811.4170 (2008)
6. Bekkali, A., Sanson, H., Matsumoto, M.: RFID Indoor Positioning Based on Probabilistic RFID Map and Kalman Filtering. In: WiMob, p. 21 (2007)
7. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
8. Cattuto, C., den Broeck, W.V., Barrat, A., Colizza, V., Pinton, J.F., Vespignani, A.: Dynamics of Person-to-Person Interactions from Distributed RFID Sensor Networks. PLoS ONE 5(7) (July 2010)
9. Cortes, C., Vapnik, V.: Support-Vector Networks. Machine Learning 20(3), 273–297 (1995)
10. Finkenzeller, K.: RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification, 2nd edn. John Wiley & Sons, Inc., New York (2003)
11. Glover, B., Bhatt, H.: RFID Essentials (Theory in Practice (O'Reilly)). O'Reilly Media, Inc., Sebastopol (2006)

12. Hightower, J., Vakili, C., Borriello, G., Want, R.: Design and Calibration of the SpotON Ad-Hoc Location Sensing System. Tech. rep. (2001)
13. Isella, L., Stehlé, J., Barrat, A., Cattuto, C., Pinton, J.F., den Broeck, W.V.: What's in a Crowd? Analysis of Face-to-Face Behavioral Networks. CoRR 1006.1260 (2010)
14. Joachims, T.: Making large-Scale SVM Learning Practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods - Support Vector Learning, ch. 11, pp. 169–184. MIT Press, Cambridge (1999)
15. Li, X., Pahlavan, K.: Super-Resolution TOA Estimation with Diversity for Indoor Geolocation. IEEE Transactions on Wireless Communications 3(1), 224–234 (2004)
16. Meriac, M., Fiedler, A., Hohendorf, A., Reinhardt, J., Starostik, M., Mohnke, J.: Localization Techniques for a Mobile Museum Information System. In: Proceedings of WCI (Wireless Communication and Information) (2007)
17. Mitchell, T.: Machine Learning (Mcgraw-Hill International Edit), 1st edn. McGraw-Hill Education, ISE Editions (October 1997)
18. Ni, L.M., Liu, Y., Lau, Y.C., Patil, A.P.: LANDMARC: Indoor Location Sensing Using Active RFID. Wireless Networks 10(6), 701–710 (2004)
19. Niculescu, D., Badrinath, B.R.: Ad Hoc Positioning System (APS) Using AOA. In: INFOCOM (2003)
20. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The Cricket Location-Support System. In: MOBICOM, pp. 32–43 (2000)
21. Rappaport, T.: Wireless Communications: Principles and Practice, 2nd edn. Prentice Hall PTR, Upper Saddle River (2001)

On the Stratification of Multi-label Data

Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas

Dept. of Informatics,
Aristotle University of Thessaloniki,
Thessaloniki 54124, Greece
{sechidis,greg,vlahavas}@csd.auth.gr

Abstract. Stratified sampling is a sampling method that takes into account the existence of disjoint groups within a population and produces samples where the proportion of these groups is maintained. In single-label classification tasks, groups are differentiated based on the value of the target variable. In multi-label learning tasks, however, where there are multiple target variables, it is not clear how stratified sampling could/should be performed. This paper investigates stratification in the multi-label data context. It considers two stratification methods for multi-label data and empirically compares them along with random sampling on a number of datasets and based on a number of evaluation criteria. The results reveal some interesting conclusions with respect to the utility of each method for particular types of multi-label datasets.

1 Introduction

Experiments are an important aspect of machine learning research [14,7]. In supervised learning, experiments typically involve a first step of distributing the examples of a dataset into two or more disjoint subsets. When training data abound, the *holdout* method is used to distribute the examples into a training and a test set, and sometimes also into a validation set. When training data are limited, *cross-validation* is used, which starts by splitting the dataset into a number of disjoint subsets of approximately equal size.

In classification tasks, the *stratified* version of these two methods is typically used, which splits a dataset so that the proportion of examples of each class in each subset is approximately equal to that in the complete dataset. Stratification has been found to improve upon standard cross-validation both in terms of bias and variance [13].

To the best of our knowledge, what stratification means for multi-label data [23] and how it can be accomplished has not been addressed in the literature. Papers conducting experiments on multi-label data use either predetermined train/test splits that come with a dataset or the random version of the holdout and cross-validation methods. Whether this version is the best that one can do in terms of variance and/or bias of estimate has not been investigated.

Furthermore, random distribution of multi-label training examples into subsets suffers from the following practical problem: it can lead to test subsets lacking even

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 145–158, 2011.

just one positive example of a rare label, which in turn causes calculation problems for a number of multi-label evaluation measures. The typical way these problems get by-passed in the literature is through complete removal of rare labels. This, however, implies that the performance of the learning systems on rare labels is unimportant, which is seldom true. As an example consider that a multi-label learner is used for probabilistic indexing of a large multimedia collection, given a small annotated sample according to a multimedia ontology. Avoiding the evaluation of the multi-label learner for rare concepts of the ontology, implies that we should not allow users to query the collection with such concepts, as the information retrieval performance level of the indexing system for these concepts would be uncertain. This limits the usefulness of the indexing system.

The above issues motivated us to investigate in this paper the concept of stratification in the context of multi-label data. Section 2 considers two interpretations of multi-label stratification. The first one is based on the distinct labelsets that are present in the dataset, while the second one considers each label independently of the rest. Section 3 proposes an algorithm for stratified sampling of multi-label data according to the second interpretation. Section 4 presents empirical results comparing the two multi-label sampling approaches as well as random sampling on several datasets in terms of a number of evaluation criteria. Results reveal some interesting relationships between the utility of each method and particular types of multi-label datasets that can help researchers and practitioners improve the robustness of their experiments. Section 5 presents the conclusions of this work and our future plans on this topic.

2 Stratifying Multi-Label Data

Stratified sampling is a sampling method that takes into account the existence of disjoint groups within a population and produces samples where the proportion of these groups is maintained. In single-label classification tasks, groups are differentiated based on the value of the target variable.

In multi-label data [23], groups could be formed based on the different *combinations of labels* (labelsets) that characterize the training examples. The number of distinct labelsets in a multi-label dataset with m examples and q labels is upper bounded by $\min(m, 2^q)$. Usually this bound equals m, because in most applications q is not very small and as a result 2^q is a very large number. Table 1 shows that, for a variety of multi-label datasets, the number of distinct labelsets is often quite large and sometimes close to the number of examples. In such cases, this *strict* interpretation of stratified sampling for multi-label data is impractical for performing k-fold cross-validation or holdout experiments, as most groups would consist of just a single example. Table 1 is actually sorted in ascending order of the ratio between distinct labelsets and number of examples and accordingly in descending order of average examples per distinct labelset. Notice that in the last two datasets, the average number of examples per labelset is 1 (rounded).

We further consider a more relaxed interpretation of stratified sampling for multi-label data, which sets as a goal the maintenance of the distribution of positive and negative examples of each label. This interpretation views each

Table 1. A variety of multi-label datasets and their statistics: number of labels, examples, distinct labelsets and distinct labelsets per example, along with the minimum, average and maximum number of examples per labelset and label

dataset	labels	examples	label sets	$\frac{\text{labelsets}}{\text{examples}}$	examples per labelset			examples per label		
					min	avg	max	min	avg	max
Scene [1]	6	2407	15	0.01	1	160	405	364	431	533
Emotions [21]	6	593	27	0.05	1	22	81	148	185	264
TMC2007 [20]	22	28596	1341	0.05	1	21	2486	441	2805	16173
Genbase [6]	27	662	32	0.05	1	21	170	1	31	171
Yeast [9]	14	2417	198	0.08	1	12	237	34	731	1816
Medical[1]	45	978	94	0.10	1	10	155	1	27	266
Mediamill [19]	101	43907	6555	0.15	1	7	2363	31	1902	33869
Bookmarks [12]	208	87856	18716	0.21	1	5	6087	300	857	6772
Bibtex [12]	159	7395	2856	0.39	1	3	471	51	112	1042
Enron[2]	53	1702	753	0.44	1	2	163	1	108	913
Corel5k [8]	374	5000	3175	0.64	1	2	55	1	47	1120
ImageCLEF2010 [16]	93	8000	7366	0.92	1	1	32	12	1038	7484
Delicious [22]	983	16105	15806	0.98	1	1	19	21	312	6495

label independently. However, note that we cannot simply apply stratification independently for each label, as this would lead to different disjoint subsets of the data for each label. Such datasets are unsuitable for evaluating multi-label learning algorithms, with the exception of the simple binary relevance approach. Even this approach, however, could only be evaluated using measures that can be calculated using independent computations for each label, such as Hamming loss and macro-averaged precision, recall and F_1.

Achieving this kind of stratification when setting up k-fold cross-validation or holdout experiments on multi-label data is meaningful, because most labels in multi-label domains are characterized by class imbalance [11,3]. The last three columns of Table 1 show the minimum, average and maximum number of examples per label for each dataset. They give an impression of the imbalance ratios found in multi-label domains.

Achieving this kind of stratification is expected to be beneficial, in two directions. Firstly, based on past studies of single-label data, it is expected to improve upon random distribution in terms of estimate bias and variance [13]. Secondly, it will lower the chance of producing subsets with zero positive examples for one or more labels. Such subsets raise issues in the calculation of certain commonly used multi-label evaluation measures, such as the macro-averaged versions of recall, F_1, area under the receiver operating characteristic curve (AUC) and average precision[3], a popular metric in multimedia information retrieval [15].

[1] http://www.computationalmedicine.org/challenge/index.php

[2] http://bailando.sims.berkeley.edu/enron_email.html

[3] The macro-averaged version of average precision is more commonly called mean average precision (MAP) in information retrieval.

Consider for example the contingency table depicted in Fig. 1, which concerns the predictions for a label. In the case where the test set has none positive examples of this label, then $fn = tp = 0$. Given that recall is defined as $tp/(tp+fn)$, the value of recall for this label is undefined (0/0). If the model is correct and doesn't predict this label for any of the test examples, then $fp = 0$, rendering the value of precision for this label undefined too (0/0), since precision is defined as $tp/(tp+fp)$. F_1 is the harmonic mean of precision and recall, which by definition is rendered undefined when one of precision and recall is undefined. AUC is also undefined, because it depends on the true positive rate, which is equivalent to recall. Average precision considers a ranking of the positively predicted examples of a label based on some confidence value. It is the average of tp precisions, $Precision_i$, $i = 1 \dots tp$, where $Precision_i$ is the precision computed for the positively predicted examples ranked higher or equally to the ith true positive example in this ranking. Since $tp = 0$, average precision is also undefined. Macro-averaging means taking the average of a measure across all labels. If a measure is undefined for one of the labels, its average across all labels is also undefined.

		predicted	
		negative	positive
actual	negative	tn	fp
	positive	fn	tp

Fig. 1. Contingency table concerning the predictions for a label

3 Iterative Stratification

We here propose an algorithm for achieving the relaxed version of multi-label stratification that we discussed in Sect. 2. The pseudo-code is given in Algorithm 1. The input to the algorithm is a multi-label data set, D, annotated with a set of labels $L = \{\lambda_1, ..., \lambda_q\}$, a desired number of subsets k and a desired proportion of examples in each subset, $r_1, \dots r_k$. For example, if we would like to use the algorithm for performing 10-fold CV, then k should be 10 and $r_1 = \dots = r_k$ should be 1/10.

The algorithm starts by calculating the desired number of examples, c_j, at each subset, S_j, by multiplying the number of examples, $|D|$, with the desired proportion for this subset r_j (lines 1-3). It then calculates the desired number of examples of each label λ_i at each subset S_j, c_j^i, by multiplying the number of examples annotated with that label, $|D^i|$, with the desired proportion for this subset r_j (lines 5-9). Note that both c_j and c_j^i will most often be decimal numbers, but this does not affect the proper functioning of the algorithm.

The algorithm is iterative (lines 10-33). It examines one label in each iteration, the one with the fewest remaining examples, denoted l (lines 13-14). The motivation for this greedy key point of the algorithm, is the following: if rare labels are not examined in priority, then they may be distributed in an undesired way, and this cannot be repaired subsequently. On the other hand with frequent

Algorithm 1. IterativeStratification(D,n, $r_1 \dots r_n$)

```
Input: A set of instances, $D$, annotated with a set of labels $L = \{\lambda_1, ..., \lambda_q\}$,
       desired number of subsets $k$, desired proportion of examples in each
       subset, $r_1, \dots r_k$ (e.g. in 10-fold CV $k = 10, r_j = 0.1, j = 1 \dots 10$)
Output: Disjoint subsets $S_1, \dots S_k$ of $D$

// Calculate the desired number of examples at each subset
for $j \leftarrow 1$ to $k$ do
    $c_j \leftarrow |D| r_j$
// Calculate the desired number of examples of each label at each subset
for $i \leftarrow 1$ to $|L|$ do
    // Find the examples of each label in the initial set
    $D^i \leftarrow \{(\mathbf{x}, Y) \in D : \lambda_i \in Y\}$
    for $j \leftarrow 1$ to $k$ do
        $c_j^i \leftarrow |D^i| r_j$
while $|D| > 0$ do
    // Find the label with the fewest (but at least one) remaining examples,
    // breaking ties randomly
    $D^i \leftarrow \{(\mathbf{x}, Y) \in D : \lambda_i \in Y\}$
    $l \leftarrow \arg\min_i(|D^i|) \bigcap \{i : D^i \neq \emptyset\}$
    foreach $(\mathbf{x}, Y) \in D^l$ do
        // Find the subset(s) with the largest number of desired examples for this
        // label, breaking ties by considering the largest number of desired examples,
        // breaking further ties randomly
        $M \leftarrow \arg\max_{j=1 \dots k}(c_j^l)$
        if $|M| = 1$ then
            $m \in M$
        else
            $M' \leftarrow \arg\max_{j \in M}(c_j)$
            if $|M'| = 1$ then
                $m \in M'$
            else
                $m \leftarrow randomElementOf(M')$
        $S_m \leftarrow S_m \bigcup \{(\mathbf{x}, Y)\}$
        $D \leftarrow D \setminus \{(\mathbf{x}, Y)\}$
        // Update desired number of examples
        foreach $\lambda_i \in Y$ do
            $c_m^i \leftarrow c_m^i - 1$
        $c_m \leftarrow c_m - 1$
return $S_1, \dots, S_k$
```

labels, we have the chance later on to modify the current distribution towards the desired, due to the availability of more examples.

Then, for each example (x, Y) of this label, the algorithm selects an appropriate subset for distribution. The first criterion for subset selection is the current desired number of examples for this label c_j^l. The subset that maximizes it gets selected (line 19). This is also a greedy choice, since this is actually the subset whose current proportion of examples of label l deviates more from the desired one. In case of ties, then among the tying subsets, the one with the highest number of desired examples c_j get selected (line 23). This is another greedy choice, since this is actually the subset whose proportion of examples irrespectively of labels deviates more from the desired one. Further ties are broken randomly (line 27).

Once the appropriate subset, m, is selected, we add the example (x, Y) to S_m and remove it from D (lines 28-29). In the end of the iteration, we decrement the number of desired examples for each label of this example at subset m, c_m^i, as well as the total number of desired examples for subset m, c_m (lines 30-33).

The algorithm will finish as soon as the original dataset gets empty. This will normally occur after $|L|$ iterations, but it may as well occur in less, due to the examples of certain labels having already been distributed. It may also occur in more, as certain datasets (e.g. mediamill) have examples that are not annotated with any label. One may argue that such examples don't carry any information, but in fact they do carry negative information for each label. These examples are distributed so as to balance the desired number of examples at each subset. This special case of the algorithm is not shown in the pseudocode of Algorithm 1 in order to keep it as legible as possible.

4 Experiments

4.1 Setup

We compare three techniques for sampling without replacement from a multi-label dataset: a) *random* sampling (R), b) stratified sampling based on distinct *labelsets* (L), as discussed in Sect. 2, and c) the *iterative* stratification technique (I), as presented in Sect. 3.

We experiment on the 13 multi-label datasets that are presented in Table 1. We have already commented on certain statistical properties of these datasets in Sect. 2. All of them, apart from ImageCLEF2010, are available for download from the web site of the Mulan library for multi-label learning[4] where their original source is also given. ImageCLEF2010 refers to the visual data released to participants in the photo annotation task of the 2010 edition of the ImageCLEF benchmark [16]. Feature extraction was performed using dense sampling with the SIFT descriptor, followed by codebook construction using k-means clustering with k=4096.

Following a typical machine learning experimental evaluation scenario, we perform 10-fold cross-validation experiments on datasets with up to 15k examples and holdout experiments (2/3 for training and 1/3 for testing) for larger

[4] http://mulan.sourceforge.net

datasets. Both types of experiments are repeated 5 times with different random orderings of the training examples. The results in the following sections are averages over these 5 runs.

4.2 Distribution of Labels and Examples

This section compares the three different sampling techniques in terms of a number of statistical properties of the produced subsets. The notation used here, follows that of Sect. 3. In particular, we consider a set of instances, D, annotated with a set of labels, $L = \{\lambda_1, ..., \lambda_q\}$, a desired number, k, of disjoints subsets of D, $S_1, \ldots S_k$, and a desired proportion of examples in each of these subsets, $r_1, \ldots r_k$. The desired number of examples at each subset S_j is denoted c_j and is equal to $|D|r_j$. The subsets of D and S_j that contain positive examples of label λ_i are denoted D^i and S_j^i respectively.

The *Labels Distribution* (LD) measure, evaluates the extent to which the distribution of positive and negative examples of each label in each subset, follows the distribution of that label in the whole dataset. For each label λ_i, the measure computes the absolute difference between the ratio of positive to negative examples in each subset S_j with the ratio of positive to negative examples in the whole dataset D, and then averages the results across all labels. Formally:

$$LD = \frac{1}{q}\sum_{i=1}^{q}\left(\frac{1}{k}\sum_{j=1}^{k}\left|\frac{|S_j^i|}{|S_j| - |S_j^i|} - \frac{|D^i|}{|D| - |D^i|}\right|\right)$$

The *Examples Distribution* (ED) measure evaluates the extend to which the number of examples of each subset S_j deviates from the desired number of examples of that subset. Formally:

$$ED = \frac{1}{k}\sum_{j=1}^{k}||S_j| - c_j|$$

For the cross-validation experiments we further compute two additional measures that quantify the problem of producing subsets with zero positive examples: a) The number of folds that contain at least one label with zero positive examples (FZ), and b) the number of fold-label pairs with zero positive examples (FLZ).

Table 2 presents the afore-mentioned statistical properties (ED, LD, FZ, FLZ) for the produced subsets in each of the 13 datasets. The best result for each dataset and measure is underlined. The second column of the table presents the ratio of labelsets to examples in each dataset to assist in the interpretation of the results that follows.

We first observe that iterative stratification achieves the best performance in terms of LD in all datasets apart from *Scene*, *Yeast* and *TMC2007*, where the labelsets-based method is better. This shows that the proposed algorithm is generally better than the others in maintaining the ratio of positive to negative examples of each label in each subset.

We further notice that the difference in LD between iterative stratification and the labelsets-based method grows with the ratio of labelsets over examples (2nd column of Table 2). Indeed, when this ratio is small (e.g. ≤ 0.1), the LD of the labelset-based method is close to that of iterative stratification, while when it is large (e.g. ≥ 0.39), it is close to that of random sampling. This behavior is reasonable, since as we discussed in Sect. 2, the larger this ratio is, the more impractical the stratification according to labelsets becomes, as each labelset annotates a very small number of examples (e.g. one or two). This also justifies the fact that the labelsets-based method managed to overcome iterative stratification in terms of LD in *Scene*, *Yeast* and *TMC2007*, as these datasets are characterized by a small ratio of labelsets over examples.

In terms of ED, the labelsets-based and the random sampling methods achieve the best performance in all datasets, while iterative stratification is much worse, with the exception of *Mediamill*. The subsets produced by these methods pay particular attention to the desired number of examples. Iterative stratification on the other hand, trades-off the requirement for constructing subsets with specified number of examples in favor of maintaining the class imbalance ratio of each label. The exception of *Mediamill* is justified from the fact that it contains a number of examples with no positive labels, which are distributed by our algorithm so as to balance the desired number of examples in each subset, as discussed in the last paragraph of Sect. 3.

Finally we observe that iterative stratification produces the smallest value for FZ and FLZ in all datasets. In the *Bibtex* and *ImageCLEF2010* datasets in particular, only iterative stratification leads to subsets with positive examples for all folds. This means that only iterative stratification allows the calculation of the multi-label evaluation measures that were mentioned in Sect. 2. All methods fail to produce subsets with positive examples for all labels in the datasets *Corel5k*, *Enron*, *Medical* and *Genbase*, which contain labels characterized by absolute rarity [11] (notice in Table 1 that the minimum number of examples per label in these datasets is just one). All methods produce subsets with at least one positive example for all labels in the *scene* and *emotions* datasets, where the minimum number of examples per label is large.

4.3 Variance of Estimates

This section examines how the variance of the 10-fold cross-validation estimates for six different multi-label evaluation measures is affected by the different sampling methods. Table 3 shows the six measures, categorized according to the required type of output from a multi-label model (two representative measures from each category). The experiments are based on the 9 out of 13 datasets, where cross-validation was applied.

Two different multi-label classification algorithms are used for performance evaluation: The popular *binary relevance* (BR) approach, which learns a single independent binary model for each label and the *calibrated label ranking* (CLR) method [10], which learns pairwise binary models, one for each pair of labels. Similarly to iterative stratification, BR treats each label independently of the

Table 2. Statistical properties of the produced subsets by a) random sampling, b) labelsets-based stratification, and c) iterative stratification: Labels Distribution (LD), Examples Distribution (ED), folds that contain at least one label with zero positive examples (FZ), and number of fold-label pairs with zero positive examples (FLZ).

dataset	$\frac{\text{labelsets}}{\text{examples}}$	stratification	ED	LD	FZ	FLZ
Scene	0.01	Random	0.42	0.0267	0 out of 10	0 out of 60
		Labelsets	0.42	0.0038	0 out of 10	0 out of 60
		Iterative	2.77	0.0043	0 out of 10	0 out of 60
Emotions	0.05	Random	0.42	0.0973	0 out of 10	0 out of 60
		Labelsets	0.42	0.0316	0 out of 10	0 out of 60
		Iterative	1.80	0.0273	0 out of 10	0 out of 60
Genbase	0.05	Random	0.32	0.0205	10 out of 10	90 out of 270
		Labelsets	0.32	0.0078	10 out of 10	77 out of 270
		Iterative	0.45	0.0055	10 out of 10	74 out of 270
TMC2007	0.05	Random	0.00	0.00250	—	
		Labelsets	0.00	0.00046		
		Iterative	27.4	0.00052		
Yeast	0.08	Random	0.42	0.0862	1 out of 10	1 out of 140
		Labelsets	0.42	0.0273	0 out of 10	0 out of 140
		Iterative	3.53	0.0342	0 out of 10	0 out of 140
Medical	0.10	Random	0.32	0.0110	10 out of 10	203 out of 450
		Labelsets	0.32	0.0059	10 out of 10	179 out of 450
		Iterative	1.47	0.0039	10 out of 10	173 out of 450
Mediamill	0.15	Random	0.33	0.00140	—	
		Labelsets	0.33	0.00056		
		Iterative	0.33	0.00002		
Bookmarks	0.21	Random	0.67	0.00026	—	
		Labelsets	0.67	0.00016		
		Iterative	71.20	0.00002		
Bibtex	0.39	Random	0.50	0.0033	1 out of 10	1 out of 1590
		Labelsets	0.50	0.0027	1 out of 10	1 out of 1590
		Iterative	7.08	0.0006	0 out of 10	0 out of 1590
Enron	0.44	Random	0.32	0.0165	10 out of 10	95 out of 530
		Labelsets	0.32	0.0132	10 out of 10	88 out of 530
		Iterative	2.96	0.0050	10 out of 10	47 out of 530
Corel5k	0.64	Random	0.00	0.0026	10 out of 10	1140 out of 3740
		Labelsets	0.00	0.0023	10 out of 10	1118 out of 3740
		Iterative	4.20	0.0010	10 out of 10	788 out of 3740
ImageCLEF2010	0.92	Random	0.00	0.0324	4 out of 10	4 out of 930
		Labelsets	0.00	0.0265	4 out of 10	4 out of 930
		Iterative	4.48	0.0069	0 out of 10	0 out of 930
Delicious	0.98	Random	0.67	0.00084	—	
		Labelsets	0.67	0.00084		
		Iterative	52.47	0.00034		

Table 3. Six multi-label evaluation measures categorized according to the required type of output from a multi-label model

Measure	Type of Output
Hamming Loss	Bipartition
Subset Accuracy	Bipartition
Coverage	Ranking
Ranking Loss	Ranking
Mean Average Precision	Probabilities
Micro-averaged AUC	Probabilities

rest. Similarly to the labelsets-based stratification, CLR considers label combinations, though only combinations of pairs of labels. Both BR and CLR are instantiated using *random forests* [2] as the binary classification algorithm underneath. We selected this particular algorithm, because it is fast and usually highly accurate without the need of careful tuning.

Following the recommendations in [5], we will discuss the results based on the average ranking of the three different stratification methods. The method that achieves the lowest standard deviation for a particular measure in a particular dataset is given a rank of 1, the next one a rank of 2 and the method with the largest standard deviation is given a rank of 3.

Table 4 shows the mean and standard deviation of the 10-fold cross-validation estimates for the six different measures on the 9 different datasets using BR, along with the average ranks: a) across datasets with small ratio of labelsets over examples (≤ 0.1), b) across datasets with large ratio of labelsets over examples (≥ 0.39), and c) across all datasets.

Looking at the last row of the table, we first notice that random sampling has the worst total average rank in all measures, as its estimates have the highest standard deviation in almost all cases. Iterative stratification has an equal or better overall rank compared to the labelsets-based method, apart from the case of Mean Average Precision. However, these ranks are computed based only on the two datasets where none of the measures was undefined. As already noted, iterative stratification manages to output an estimate in two datasets more than the labelsets-based method and three datasets more than random sampling.

We then look at the average ranks for the upper and lower part of the table that differ in terms of the labelsets over examples ratio. We notice that in the upper part of the table, the labelsets-based method exhibits better rank in all measures, apart from ranking loss. On the other hand, in the lower part of the table, iterative stratification is better than the other methods for all measures. This reinforces the conclusion of the previous section, where we found that the labelsets-based method is more suited to datasets with small ratio of labelsets over examples.

As far as the measures are concerned, we notice that iterative stratification is particularly well suited to ranking loss, independently of the ratio of labelsets over examples. This may seem strange at first sight, as ranking loss is a measure

Table 4. Mean and standard deviation of six multi-label evaluation measures (columns 3 to 8) computed using 10-fold cross validation, the binary relevance algorithm and the three different sampling methods: (R)andom, (L)abelsets, and (I)terative. The first 5 rows correspond to datasets with small ratio of labelsets over examples (≤ 0.1), followed by the average rank of each method. The next 4 rows correspond to datasets with large ratio of labelsets over examples (≥ 0.39), followed by the average rank of each method. The last line presents the average rank for all 9 datasets.

dataset	str.	Hamming Loss	Subset Accuracy	Coverage	Ranking Loss	Mean Average Precision	Micro-averaged AUC
Scene	R	0.0806±0.0078	0.5938±0.0333	0.3542±$\underline{0.0406}$	0.0543±0.0070	0.8695±0.0177	0.9612± 0.0064
	L	0.0801±$\underline{0.0059}$	0.5959±0.0279	0.3557±0.0421	0.0545±0.0082	0.8696±0.0163	0.9616± $\underline{0.0055}$
	I	0.0805±0.0060	0.5947±$\underline{0.0261}$	0.3573±0.0454	0.0549±$\underline{0.0069}$	0.8699± $\underline{0.0149}$	0.9613±0.0058
Emotions	R	0.1809±0.0193	0.3247±0.0570	1.6528±0.1424	0.1397±0.0269	0.7568±0.0378	0.8777±0.0197
	L	0.1792±$\underline{0.0170}$	0.3299±$\underline{0.0434}$	1.6394±$\underline{0.1221}$	0.1367±$\underline{0.0223}$	0.7603±$\underline{0.0340}$	0.8804±$\underline{0.0193}$
	I	0.1786±0.0175	0.3270±0.0553	1.6453±0.1308	0.1380±0.0265	0.7616±0.0409	0.8787±0.0222
Genbase	R	0.0024±0.0013	0.9444±0.0295	0.4077±0.2308	0.0030±0.0043	NaN±NaN	0.9952±0.0075
	L	0.0024±0.0012	0.9444±0.0267	0.3995±0.1968	0.0027±0.0038	NaN±NaN	0.9957±0.0063
	I	0.0024±$\underline{0.0011}$	0.9438±$\underline{0.0232}$	0.3878±$\underline{0.1808}$	0.0025±$\underline{0.0032}$	NaN±NaN	0.9962±$\underline{0.0054}$
Yeast	R	0.1892±0.0070	0.1746±0.0196	6.1383±0.1853	0.1584±0.0108	NaN±NaN	0.8533±0.0093
	L	0.1884±$\underline{0.0045}$	0.1762±$\underline{0.0146}$	6.1236±$\underline{0.1082}$	0.1576±$\underline{0.0063}$	0.5358±$\underline{0.0160}$	0.8543±$\underline{0.0062}$
	I	0.1887±0.0051	0.1757±0.0185	6.1247±0.1219	0.1578±0.0076	0.5429±0.0198	0.8539±0.0071
Medical	R	0.0153±0.0014	0.4531±0.0413	1.5570±0.4584	0.0224±0.0071	NaN±NaN	0.9789±0.0072
	L	0.0151±$\underline{0.0012}$	0.4616±$\underline{0.0351}$	1.5022±0.3972	0.0217±0.0071	NaN±NaN	0.9798±0.0062
	I	0.0151±0.0014	0.4557±0.0400	1.4497±$\underline{0.3715}$	0.0209±$\underline{0.0069}$	NaN±NaN	0.9803±$\underline{0.0058}$
Average Rank (≤ 0.1)	R	2.9	3	2.6	2.7	2.5	2.8
	L	1.2	1.4	1.6	1.9	1.5	1.4
	I	1.9	1.6	1.8	1.4	2	1.8
Bibtex	R	0.0308±0.0029	0.1015±0.0101	44.9221±1.5618	0.2130±0.0083	NaN±NaN	0.7780±0.0066
	L	0.0315±0.0021	0.1025±$\underline{0.0064}$	45.0660±$\underline{1.0094}$	0.2140±$\underline{0.0066}$	NaN± NaN	0.7682±0.0056
	I	0.0313±$\underline{0.0017}$	0.1029±0.0079	44.6686±1.0397	0.2181±0.0067	0.3505±$\underline{0.0100}$	0.7691±$\underline{0.0054}$
Enron	R	0.0475±0.0020	0.1229±$\underline{0.0179}$	12.7126±1.0364	0.0820±0.0084	NaN±NaN	0.9138±0.0068
	L	0.0474±0.0021	0.1245±0.0202	12.6388±0.8306	0.0810±0.0070	NaN±NaN	0.9148±0.0074
	I	0.0474±$\underline{0.0018}$	0.1213±0.0197	12.4571±$\underline{0.6189}$	0.0797±$\underline{0.0062}$	NaN±NaN	0.9165±$\underline{0.0055}$
Corel5k	R	0.0094±$\underline{0.0001}$	0.0032±0.0023	217.6020±5.5919	0.2717±0.0084	NaN±NaN	0.7821±0.0061
	L	0.0094±$\underline{0.0001}$	0.0022±$\underline{0.0020}$	217.1086±4.7339	0.2708±0.0086	NaN±NaN	0.7827±0.0060
	I	0.0094±$\underline{0.0001}$	0.0026±0.0022	217.4484±$\underline{4.0884}$	0.2701±$\underline{0.0058}$	NaN±NaN	0.7834±$\underline{0.0044}$
Image CLEF2010	R	0.0996±0.0013	0.0003±0.0006	60.5913±0.8638	0.1391±0.0025	NaN±NaN	0.8591±0.0023
	L	0.0997±0.0013	0.0005±0.0007	60.6276±0.8242	0.1392±0.0022	NaN±NaN	0.8589±0.0021
	I	0.0997±$\underline{0.0008}$	0.0001±$\underline{0.0004}$	60.8236±$\underline{0.6342}$	0.1394±$\underline{0.0021}$	0.2338±$\underline{0.0048}$	0.8588±$\underline{0.0019}$
Average Rank (≥ 0.39)	R	2.4	2.3	3.0	2.8		2.8
	L	2.4	2.0	1.8	2.0	-	2.3
	I	1.3	1.8	1.3	1.3		1.0
Average Rank	R	2.7	2.7	2.8	2.7	2.5	2.8
	L	1.7	1.7	1.7	1.9	1.5	1.8
	I	1.6	1.7	1.6	1.3	2.0	1.4

computed across all labels for a given test example. However, it is also true that good ranking loss for BR depends on good probability estimates for each label, which in turn is affected by the distribution of positive and negative examples for each label.

Table 5 shows the mean and standard deviation of the 10-fold cross-validation estimates for the six different measures using CLR on 5 datasets only, those with less than 50 labels, as the quadratic space complexity of CLR resulted into memory shortage problems during our experiments with datasets having more than 50 labels. The last row shows the average rank of the three stratification methods across these datasets.

We here notice that random sampling again has the worst average rank, while the labelsets-based method is better than iterative stratification, even in terms of ranking loss. In this experiment, all datasets have a small ratio of labelesets over examples (≤ 0.1), so according to what we have seen till now, the behavior that we notice is partly expected.

Table 5. Mean and standard deviation of six multi-label evaluation measures (columns 3 to 8) computed using 10-fold cross validation, the calibrated label ranking (CLR) algorithm and the three different sampling methods: (R)andom, (L)abelsets, and (I)terative. The last row shows the average rank of the three stratification methods across the datasets.

dataset	str.	Hamming Loss	Subset Accuracy	Coverage	Ranking Loss	Mean Average Precision	Micro-averaged AUC
Scene	R	0.0807±0.0073	0.5899±0.0329	0.3943±0.0498	0.0624±0.0085	0.8246±0.0242	0.9423±0.0082
	L	0.0802±0.0051	0.5918±0.0237	0.3884±0.0358	0.0613± 0.0070	0.8137±0.0218	0.9427±0.0062
	I	0.0808±0.0061	0.5898±0.0267	0.3891±0.0534	0.0612±0.0082	0.8191±0.0232	0.9423±0.0083
Emotions	R	0.1803±0.0196	0.3264±0.0575	1.6522±0.1311	0.1400±0.0255	0.7240±0.0500	0.8583±0.0225
	L	0.1795±0.0169	0.3272±0.0384	1.6354±0.1260	0.1365± 0.0221	0.7230±0.0371	0.8603±0.0202
	I	0.1782±0.0171	0.3278±0.0553	1.6457±0.1261	0.1382±0.0237	0.7405±0.0432	0.8596±0.0221
Genbase	R	0.0024±0.0013	0.9444±0.0295	0.4743±0.2597	0.0043±0.0049	NaN±NaN	0.9907±0.0083
	L	0.0025±0.0012	0.9432±0.0270	0.4651±0.2040	0.0041±0.0041	NaN±NaN	0.9913±0.0064
	I	0.0024±0.0012	0.9438±0.0238	0.4879±0.1925	0.0045±0.0037	NaN±NaN	0.9898±0.0063
Yeast	R	0.1888±0.0071	0.1756±0.0183	6.0975±0.1887	0.1580±0.0109	NaN±NaN	0.8339±0.0095
	L	0.1883±0.0045	0.1793±0.0143	6.0852±0.1050	0.1570± 0.0062	0.4830±0.0134	0.8346±0.0058
	I	0.1883±0.0052	0.1791±0.0198	6.0895±0.1109	0.1575±0.0069	0.5670±0.0254	0.8344± 0.0059
Medical	R	0.0154±0.0014	0.4497±0.0402	2.2879±0.5601	0.0337±0.0075	NaN±NaN	0.9610±0.0100
	L	0.0150±0.0012	0.4612±0.0359	2.1629±0.3853	0.0319±0.0069	NaN±NaN	0.9631±0.0073
	I	0.0152±0.0013	0.4532±0.0398	2.1774±0.2709	0.0320±0.0052	NaN±NaN	0.9629±0.0052
Average Rank	R	3.0	2.8	2.8	3.0	3.0	2.8
	L	1.1	1.2	1.4	1.4	1.0	1.4
	I	1.9	2.0	1.8	1.6	2.0	1.8

However, if we compare the rankings in Table 5 with the rankings in the upper part of Table 4, which contains exactly the same datasets, we notice that for CLR the benefits of the labesets-based method are larger. We attribute this to the fact that contrary to BR, CLR does consider combinations between pairs of labels, and contrary to iterative stratification, the labelsets-based method distributes examples according to label combinations.

It is also interesting to notice that the measure where iterative stratification exhibits the best performance is again ranking loss, as in the case of BR.

5 Conclusions and Future Work

This paper studied the concept of stratified sampling in a multi-label data context. It presented two different approaches for multi-label stratification and empirically investigated their performance in comparison to random sampling on several datasets and in terms of several criteria.

The main conclusion of this work can be summarized as follows:

- Labelsets-based stratification achieves low variance of performance estimates for datasets where the ratio of distinct labelsets over examples is small, irrespectively of the learning algorithm. It also works particularly well for the calibrated label ranking algorithm. This could be generalizable to other algorithms that take into account label combinations.
- Iterative stratification approach achieves low variance of performance estimates for datasets where the ratio of the distinct labelsets to the number of examples is large. This was observed when the binary relevance approach was used, but could be generalizable to other algorithms, especially those learning a binary model for each label in one of their steps [18,4]. Furthermore,

iterative stratification works particularly well for estimating the ranking loss, independently of algorithm and dataset type. Finally, iterative stratification produces the smallest number of folds and fold-label pairs with zero positive examples and it manages to maintain the ratio of positive to negative examples of each label in each subset.
- Random sampling is consistently worse than the other two methods and should be avoided, contrary to the typical multi-label experimental setup found in the literature.

In this paper we mainly focused on the application of stratified sampling to experimental machine learning, in particular producing subsets for cross-validation and holdout experiments. Apart from the purpose of estimating performance, cross-validation and holdout are also widely used for hyper-parameter selection, model selection and overfitting avoidance (e.g. reduced error pruning of decision trees/rules). The points of this paper are relevant for all these applications of stratified sampling in learning from multi-label data. For example, the stratified sampling approaches discussed in this paper could be used for reduced error pruning of multi-label decision trees [24], for down-sampling without replacement in the ensembles of pruned sets approach [17] and for deciding when to stop the training of a multi-label neural network [25].

In the future, we plan to investigate the construction of a hybrid algorithm that will combine the benefits of both the iterative and the labelsets-based stratification, in order to have a single solution that will work well for any type of dataset, classification algorithm and evaluation measure.

References

1. Boutell, M., Luo, J., Shen, X., Brown, C.: Learning multi-label scene classification. Pattern Recognition 37(9), 1757–1771 (2004)
2. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
3. Chawla, N.V., Japkowicz, N., Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. SIGKDD Explorations 6(1), 1–6 (2004)
4. Cheng, W., Hüllermeier, E.: Combining instance-based learning and logistic regression for multilabel classification. Machine Learning 76(2-3), 211–225 (2009)
5. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
6. Diplaris, S., Tsoumakas, G., Mitkas, P., Vlahavas, I.: Protein classification with multiple algorithms. In: Bozanis, P., Houstis, E.N. (eds.) PCI 2005. LNCS, vol. 3746, pp. 448–456. Springer, Heidelberg (2005)
7. Drummond, C.: Machine learning as an experimental science (revisited). In: 2006 AAAI Workshop on Evaluation Methods for Machine Learning, pp. 1–5 (2006)
8. Duygulu, P., Barnard, K., de Freitas, J.F.G., Forsyth, D.A.: Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2353, pp. 97–112. Springer, Heidelberg (2002)
9. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in Neural Information Processing Systems, vol. 14 (2002)

10. Fürnkranz, J., Hüllermeier, E., Mencia, E.L., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning 73(2), 133–153 (2008)
11. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering 21, 1263–1284 (2009)
12. Katakis, I., Tsoumakas, G., Vlahavas, I.: Multilabel text classification for automated tag suggestion. In: Proceedings of the ECML/PKDD 2008 Discovery Challenge, Antwerp, Belgium (2008)
13. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: IJCAI, pp. 1137–1145 (1995)
14. Langley, P.: Machine learning as an experimental science. Machine Learning 3, 5–8 (1988)
15. Nowak, S., Lukashevich, H., Dunker, P., Rüger, S.: Performance measures for multilabel evaluation: a case study in the area of image classification. In: Proceedings of the International Conference on Multimedia Information Retrieval, MIR 2010, pp. 35–44. ACM, New York (2010)
16. Nowak, S., Huiskes, M.: New strategies for image annotation: Overview of the photo annotation task at imageclef 2010. Working Notes of CLEF 2010 (2010)
17. Read, J., Pfahringer, B., Holmes, G.: Multi-label classification using ensembles of pruned sets. In: Proc. 8th IEEE International Conference on Data Mining (ICDM 2008), pp. 995–1000 (2008)
18. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: Proc. 20th European Conference on Machine Learning (ECML 2009), pp. 254–269 (2009)
19. Snoek, C.G.M., Worring, M., van Gemert, J.C., Geusebroek, J.M., Smeulders, A.W.M.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: MULTIMEDIA 2006: Proceedings of the 14th Annual ACM International Conference on Multimedia, pp. 421–430. ACM, New York (2006)
20. Srivastava, A., Zane-Ulman, B.: Discovering recurring anomalies in text reports regarding complex space systems. In: Proc. 2005 IEEE Aerospace Conference, pp. 3853–3862 (2005)
21. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In: Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA (2008)
22. Tsoumakas, G., Katakis, I., Vlahavas, I.: Effective and efficient multilabel classification in domains with large number of labels. In: Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD 2008), pp. 30–44 (2008)
23. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, 2nd edn., ch. 34, pp. 667–685. Springer, Heidelberg (2010)
24. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning 73(2), 185–214 (2008)
25. Zhang, M.L., Zhou, Z.H.: Multi-label neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering 18(10), 1338–1351 (2006)

Learning First-Order Definite Theories via Object-Based Queries

Joseph Selman and Alan Fern

School of Electrical Engineering and Computer Science,
Oregon State University
{selman,afern}@eecs.oregonstate.edu

Abstract. We study the problem of exact learning of first-order definite theories via queries, toward the goal of allowing humans to more efficiently teach first-order concepts to computers. Prior work has shown that first order Horn theories can be learned using a polynomial number of membership and equivalence queries [6]. However, these query types are sometimes unnatural for humans to answer and only capture a small fraction of the information that a human teacher might be able to easily communicate. In this work, we enrich the types of information that can be provided by a human teacher and study the associated learning problem from a theoretical perspective. First, we consider allowing queries that ask the teacher for the relevant objects in a training example. Second, we examine a new query type, called a pairing query, where the teacher provides mappings between objects in two different examples. We present algorithms that leverage these new query types as well as restrictions applied to equivalence queries to significantly reduce or eliminate the required number of membership queries, while preserving polynomial learnability. In addition, we give learnability results for certain cases of imperfect teachers. These results show, in theory, the potential for incorporating object-based queries into first-order learning algorithms in order to reduce human teaching effort.

1 Introduction

This work is motivated by the goal of enabling non-experts in first-order logic to easily and efficiently teach first-order concepts to computers. We are interested in situations where the teacher has a reasonable semantic understanding of the target concept but is unable to provide a definition due to lack of expertise in formal knowledge representation. A key research issue is to understand the types of information that a teacher can provide and how to best use them for learning. Most work on learning first-order concepts limits the teacher to labeling examples as positive or negative, which is a restricted and indirect form of teaching.

In this work, we study algorithms that elicit and utilize information about the relevance and correspondence between objects in first-order training examples. Given that examples are often produced and/or analyzed by teachers, it is reasonable to expect that teachers will be able to at least partially provide this

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 159–174, 2011.

information. The question we ask is: How can we use such information and what will its impact on learning efficiency be?

We study this question from a theoretical perspective where the problem is to exactly identify a target first-order definite theory using labeled examples and queries to a teacher. Prior theoretical work [10,6] focused on using equivalence queries to obtain positive and negative examples, and on membership queries to obtain example labels from the teacher. While the number of queries is shown to be polynomial in certain problem parameters, the use of only these queries is quite restricted and membership queries can be problematic in practice. For example, an algorithm may ask membership queries about examples that will appear nonsensical to a typical teacher which is a well-documented problem [4]. In general, past work has not given much consideration to how suitable such queries might be for humans to answer.

We consider the use of two new types of queries that are directly about the objects in examples provided by the teacher. As such, it is plausible that the queries might be more easily answered than queries about synthetic examples constructed by the learner. The first query type is a relevant object query that returns the set of "relevant objects" in a specific example. We show that by using this query type it is possible to significantly reduce the number of required membership queries. Further, we show that membership queries can be reduced or eliminated altogether with simple restriction applied to the equivalence oracle. We also analyze what happens when the teacher's responses are imperfect, showing that there can still be benefits if the imperfection is bounded. The second query type considered is the use of pairing queries, which ask the teacher to match "corresponding" objects between two specific examples. We show that with such queries membership queries can be eliminated.

Our main contribution is a theoretical investigation into the use of object-based queries for learning first-order concepts. To our knowledge this is the first such investigation. As our results are of a worst-case nature, the exact bounds and algorithms are likely not directly applicable to practice. However, similar to past work on identifying Horn theories [6], which led to new practical algorithms [3], we expect that our work will lead to practical algorithms for utilizing object-based feedback from teachers.

2 Previous Work

The problem of exactly learning propositional Horn sentences from membership and equivalence queries was shown to have polynomial runtime with a fixed number of variables and a fixed number of clauses [2]. This algorithm was later generalized to learning first-order Horn theories from equivalence and membership queries for several learning settings, including learning from interpretations and learning from entailment [6]. All of the algorithms in this paper are built upon the algorithms given in [6].

A different algorithm using the same query types was also shown to have polynomial runtime with certain fixed problem parameters for first-order Horn theories where all the consequents involve the same predicate (called Horn programs) [10]. The generalization process used least-general-generalizations with a much finer minimization step than used here. This algorithm was used to learn control knowledge in a planning setting using goal-decomposition rules [11].

The notion of using relevant objects provided by a teacher to initialize a search-based first-order rule learner was examined [13]. Similarly, a system that used "nudges" and object highlighting to learn various constraints was created for programming games [7]. Both approaches incorporate information about objects from the teacher into experimental systems, but neither provide a theoretical analysis of the learning problem, which is the main emphasis of our work.

Recently various error models were introduced for membership queries [5] and the resulting behavior was analyzed, similar to the (j, f)-based queries discussed here. A different type of algorithm analysis that accounts for noise in learning theory is the PAC-Learning framework introduced by [12]. With query learning, it was shown that any language exactly learnable by equivalence queries is also PAC-Learnable [1]. For examples of first-order logic PAC-Learning algorithms that learn with certain fixed problem parameters, see [9], [11] and [6].

3 Preliminaries

We assume a fixed set of first-order predicates P, each with an associated arity that is upper-bounded by a. An *atom* is a predicate from P applied to the proper number of variables or constants. A *literal* is either an atom (positive literal) or the negation of an atom (negative literal). A *clause* is a disjunction of literals and is a *ground clause* if it contains no variables. The *head* of a clause is its set of positive literals and the *body* is the set of negative literals. A *definite clause* is one that contains exactly one positive, or head, literal. A set of definite clauses is a definite theory. We will be interested in the space of all definite theories over P that do not contain constants, denoted by $\mathcal{H}_D(P)$.

Following prior work [6] and without loss of generality, our algorithms employ the notion of d-subsumption (also known as object-identity subsumption) between clauses. A clause C_1 *d-subsumes* clause C_2 if there exists a substitution θ, where no two variables map to the same object, such that $C_1\theta \subseteq C_2$ (viewing clauses as sets of literals). That is, the substitution must be a 1-1 mapping.

Problem Setup. Our goal is to exactly learn an unknown target hypothesis $T \in \mathcal{H}_D(P)$ by interacting with different types of oracles. These oracle types can be thought of as different ways of interacting with a teacher. In particular we are interested in how many queries must be issued to the oracles in order to learn a hypothesis $H \in \mathcal{H}_D(P)$ that is equivalent to T with respect to coverage of examples as defined below.

In our formulation a *training example* is a definite ground clause. The body of an example can be viewed as a set of facts describing the context of an example and the single head literal can be viewed as a predicate that we would like to predict given the context. Readers familiar with inductive logic programming can view the bodies of our examples as encoding the ground background knowledge that is relevant to a particular example. For example, if the problem was to learn what configurations of a chessboard represent a state where a king is in check, the body might consist of literals describing the state of the board and the head would be a literal representing the concept of check. Note that the head predicate is not restricted to be the same across all examples.

A theory $T \in \mathcal{H}_D(P)$ *covers* an example E if there is at least one clause in T that d-subsumes E. A *positive example* is one that is covered by the target theory, while all other examples are *negative*. Note that, as detailed in prior work [6], there is no loss in generality in defining coverage in terms of d-subsumption rather than standard subsumption. In particular, for any target theory T defined relative to standard subsumption, there is an equivalent theory (possibly larger) under d-subsumption. Note that under our definition, positive examples are required to be directly covered by at least one clause in the target hypothesis. This is a weaker notion of coverage than entailment, since examples that are entailed may not be considered covered according to subsumption. For example, a theory with the two clauses $P(X) \rightarrow Q(X)$ and $Q(X) \rightarrow R(X)$, entails the example $P(c) \rightarrow R(c)$, but does not cover the example under subsumption. This difference does not limit the learnability of $\mathcal{H}_D(P)$, rather it will influence the behavior of the oracles when generating examples and answering queries.

Standard Query Types. The two most common query types in learning theory are equivalence queries and membership queries. They have been used exclusively in past work on learning first-order definitions. Equivalence queries provide a way for algorithms to obtain new counter-examples for the current hypothesis (examples where the hypothesis and target disagree about coverage). Note that in practice humans are not expected to actually answer equivalence queries, since this would require a full understanding of the current hypothesis and target concept, which is clearly impractical. Rather, equivalence queries are best viewed as a theoretical model for a large set of labeled training examples, which can produce counter examples. In this sense, a practical implementation of any of the algorithms described below would implement an equivalence oracle via such a training set that is automatically checked for consistency with the current hypothesis. This type of equivalence oracle implementation has been used successfully in the past for moving from a theoretical algorithm to practice [3]. Formally, in our setting we have:

Definition 1. *An* equivalence query (EQ) *takes a definite theory H as input and returns "done" if H is equivalent to the target theory T with respect to example coverage. Otherwise, a counter-example is returned.*

Membership queries allow an algorithm to generate an example and ask the oracle to label it. There are no restrictions placed on examples used for MQs which is a known problem [4] with human teachers, as the learner can generate nonsensical examples that confuse the teacher.

Definition 2. *A* membership query (MQ) *takes an example E as input and returns "true" if E is a positive example of the target T. Otherwise "false" is returned.*

Our results will provide worst-case bounds on the number of queries (of various types) needed to exactly learn a target theory. These bounds will depend on certain quantities related to the target theory. Table 1 provides a reference for symbols used to represent these quantities.

Table 1. Notational reference

Variable	Meaning
T	target definite theory
P	fixed set of predicates
a	largest arity of any predicates in P
k	largest number of distinct variables in a clause of T
m	number of clauses in T
n	largest number of objects in any example

4 Base Algorithm

Our algorithms are based on Khardon's algorithmic framework for learning first-order Horn theories from membership and equivalence queries [6]. As a starting point, we describe a base algorithm for our problem that uses membership and equivalence queries. This algorithm is similar to those in [6] but adapted to the specific learning problem outlined above.[1] Our later algorithms are based on viewing membership queries as the "assembly language of learning queries", which are grouped into functional units and replaced by higher-level object-based queries. As we will see, many of the queries used in the base algorithm can be reformulated as queries about objects.

Overview. Algorithm Learn-MQ gives pseudo-code for our base algorithm. The algorithm maintains a current definite theory hypothesis H and a set of ground clauses S from which H is derived. The algorithm initializes S and H to the empty sets and then enters the main loop which updates S and H on each iteration. Each loop iteration begins by issuing an equivalence query. If the result is "done", then H is correct and is returned as the answer. If a positive counter example is returned (we will see that this is always the case for perfect oracles), then MQs are used to incorporate the positive example into the current hypothesis which results in a current clause being generalized or a new clause being added. Note that there is no if condition for a negative counter example; it will become apparent below that our algorithm will never overgeneralize, making negative counter examples unnecessary.

Positive examples are incorporated into H via two steps. First, the procedure call Min-MQ$(E, \emptyset)$, described below, returns an example $E' \subseteq E$ that is still

[1] Khardon's main algorithm was for the setting of learning from interpretations. It was then shown how to place a wrapper around this algorithm for learning from entailment, which most resembles our formulation here. The algorithm described here is defined directly for our learning problem rather than taking a wrapper approach and hence is novel, though the key ideas follow rather directly from Khardon's work.

a positive example of the target theory, but where literals involving "irrelevant objects" have been removed. Here irrelevant objects are those that are not necessary in finding a variable substitution showing a target clause covers E. Given the minimized example, the algorithm then calls Pair-MQ(E', S), described below, in order to merge E' with the set of ground clauses in S. The set S is maintained so that distinct ground clauses correspond to distinct clauses in the target theory. The example E' is combined with an existing clause $s \in S$ if E' and s are covered by a common target clause. Otherwise, if no such s exists then E' is added as a new element to S. Given the updated S, the hypothesis H is set to a variabilized version of S (via the call **variabilize**(s), where distinct constants in s are replaced with distinct variables). Below we review the key functions Min-MQ and Pair-MQ and show the correctness and worst-case number of queries for Learn-MQ.

```
$S = \emptyset$, $H = \emptyset$
repeat
    $E = EQ(H)$
    if $E = done$ then return $H$
    if $E$ is a positive counter example then
        $E' =$ Min-MQ$(E, \emptyset)$                                  // See text
        $S =$ Pair-MQ$(E', S)$
    $H =$ variabilize$(S)$
```

Algorithm Learn-MQ. Learns $\mathcal{H}_D(P)$ using EQ and MQ queries.

```
Data: An example $E'$ and a set of positive examples $S$
Result: $S$ is updated by appending $E'$ or by modifying $S$
for each $s \in S$ do
    for every pairing $J$ between $s$ and $E'$ do
        if MQ$(J)$ then
            return $S = S \cup \{J\} - s$
return $S \cup \{E\}$
```

Algorithm Pair-MQ. Updates the set of examples S with a new example E' using MQs

Minimizing Examples. Algorithm Min-MQ (omitted due to space constraints) accepts an example and a list of objects already known to be relevant and considers each object o in E (and not in the known relevant list) and removes literals involving objects that are determined to be irrelevant. To test the relevance of object o, all literals in the current E involving o are dropped and a MQ is issued on the resulting example. If the MQ indicates that the result is still a positive example, then o is considered irrelevant. If the example becomes negative, then we know that o must bind with some variable in a clause of the target theory T so we should not remove it. Importantly, after minimization an example is guaranteed to be positive and will have at most k objects in it, where k is the largest number of variables contained in a single clause of T.

Merging via Pairing Examples. Algorithm Pair-MQ illustrates how a minimized example E' is combined with the current set of ground clauses S. A key operation in the algorithm is *pairing* two examples E_1 and E_2 to produce a new example J as follows: 1) Select a 1-to-1 mapping M between the objects in E_1 and E_2, 2) Let E_1' be a version of E_1 with its objects mapped according to M and return $J = E_1' \cap E_2$.

Example 1. The pairing of

$$E_1 = P(1,2), P(2,3), R(1,1) \rightarrow Q(1,3)$$

and

$$E_2 = P(a,b), P(b,c), R(b,b) \rightarrow Q(a,c)$$

under mapping $\{1/a, 2/b, 3/c\}$ results in:

$$J = P(a,b), P(b,c) \rightarrow Q(a,c).$$

Although the constants from E_2 are used in J, they are used in our algorithms as placeholders for variables.

In general, there will be exponentially many pairings in the number of objects in the examples (which is bounded by k in the algorithm). Note that this operation is similar to the standard least-general generalization (LGG) operator [8] (the LGG can be viewed as the conjunction of all pairings), but unlike the LGG operation, the result is guaranteed to be no larger than the input examples.

The algorithm attempts to find an example $s \in S$ that is covered by a target clause C that also covers E'. It then generalizes s and E' to a new clause J that is a (smaller) positive example that is also covered by C. In this way, examples in S are continually being generalized and the number of examples in S is bounded until they become maximally general. This process is carried out by considering each possible pairing between E' and s and then asking an MQ about the resulting example J. If J is positive, then it replaces s in S with J. This is justified by the following:

Lemma 1. *For any examples E_1 and E_2, we have that E_1 and E_2 are covered by a common target clause C if and only if there exists a pairing J of E_1 and E_2 that is covered by C.*

The proof is omitted due to space constraints, but it is similar to the corresponding result in [6].

Note that, according to this result, the number of examples in S will only grow when a new positive example E is not covered by any of the target clauses that cover current examples in S. Thus, the size of S will never grow larger than the number of target clauses.

Correctness and Complexity. We now show that Learn-MQ makes progress after each EQ resulting in a bound on the number of queries overall. We first give two simple lemmas.

Lemma 2. *If algorithm Pair-MQ replaces an example $s \in S$ with a new example J, then J contains strictly fewer literals than s.*

Proof. J can never grow larger than s by the definition of pairing. Suppose J and s were the same size. This means that no literals were dropped during the pairing of E and s for some 1-to-1 mapping. This implies that the variabilized version of s in H covers E', which implies that H covers E. However, this is a contradiction, since E' is guaranteed by the main loop of Learn-MQ to be a positive counter example to H. □

Lemma 3. *The size of S in algorithm Learn-MQ will never grow larger than the number of clauses in the target hypothesis.*

Proof. Suppose that $|S| > |T|$. As all examples in S are positive, there must be two examples s_1 and s_2 in S that are covered by the same clause C in T. By Lemma 1, there is a pairing of s_1 and s_2 that is also covered by C. However, we know that one of s_1 or s_2 was added to S in the presence of the other and that would only be done if there was no such pairing, showing a contradiction. □

We can now give the main result, showing that for constant a (max predicate arity) and k (max variables per clause) exact learning of $\mathcal{H}_D(P)$ is possible with a polynomial number of queries.

Theorem 1. *For any $T \in \mathcal{H}_D(P)$, Algorithm Learn-MQ learns an equivalent hypothesis after at most $|P|mk^a$ equivalence queries and $|P|mk^a(n+mk^k)$ membership queries.*

Proof. (sketch) The maximum size of an example (as measured by the number of literals) is $|P|n^a$. For every example, the algorithm removes an object and then asks a membership query. Therefore, the number of membership queries used on minimization for each example is bound by n.

After an example is minimized the maximum size of E' is $|P|k^a$, as there are at most k objects in a binding to a clause C in T. The algorithm now searches all mappings of objects to find a correct pairing. For two examples, both with k objects, the number of mappings is k^k. In the worst case, we will search all mappings for all elements in S (limited by m as shown by Lemma 3). Therefore the number of membership queries used for pairing an example is bound by mk^k.

Because every pairing removes at least one literal from an element in S, by Lemma 2 the number of examples returned by EQs is bound by $|P|mk^a$, as it is easily verified that EQs will only return positive counter-examples.

Finally, the number of membership queries is bound by $|P|mk^a(n+mk^k)$. □

5 Object-Based Queries

While EQs and MQs have been shown to be sufficient for efficient learning of $\mathcal{H}_D(P)$ with constant k and a, there is no reason to believe that such query types are ideal for use with human teachers. In particular, MQs are arguably the most primitive form of query and provide a relatively small amount of information compared to the teacher effort required to answer them (this involves analyzing a novel example which may or may not make sense from a semantic perspective).

Here we focus on reducing the number of membership queries required for learning by allowing for queries about the relevance of objects in an example. It is reasonable to expect that natural interfaces for such queries can be constructed in many teaching domains, where the teacher is able to mark the relevant objects in examples that they have already created or analyzed. For example, if the goal was to teach the legal moves of chess, the interface would allow for the selection of pieces and squares on a chess board.

5.1 Relevant Object Queries

In Min-MQ, membership queries are used to discover the objects necessary for a positive example to remain covered by the target. Since this information is likely obvious to a human teacher in many situations, it makes sense to consider a new type of query that directly asks for the same information.

Definition 3. *A* relevant object query (ROQ) *takes a positive example E as input and returns a minimal set of objects Q bound in a substitution θ such that for some clause $C \in T$, $C\theta \subseteq E$.*

Note that when an example is covered by multiple target clauses, there is not a unique answer to an ROQ. Given a set of objects Q and an example E, we let $E[Q]$ denote a new example that discards any literal in E that involves an object outside of Q. It is easy to verify that if a set of objects Q' is a strict subset of ROQ(E), then $E[Q']$ will not be covered by the target theory. Thus, a single ROQ can play the same role as Min-MQ in the base algorithm. This yields the most basic modification to Learn-MQ called *Learn-MQ-ROQ* that uses ROQs for example minimization instead of Min-MQ. In particular, Learn-MQ-ROQ is identical to Learn-MQ except that a call to Min-MQ for example E is replaced with E[ROQ(E)]. This results in a decrease in the number of MQs with the addition of ROQs.

Proposition 1. *For any $T \in \mathcal{H}_D(P)$, Learn-MQ-ROQ learns an equivalent hypothesis after at most $|P|mk^a$ equivalence queries and ROQ queries, and reduces the number of membership queries over Learn-MQ by $|P|mk^a n$. Learn-MQ-ROQ does not depend on the maximum number of objects per example n.*

If, on average, the teacher cost of answering ROQs is less than a factor of n more than answering MQs, then there is an overall reduction in the teaching cost. This is a plausible situation, particularly in domains with many objects where a human teacher can often easily ignore the potentially large number of irrelevant objects.

5.2 Eliminating Membership Queries

With the aid of relevant object queries, one might wonder if it is possible to eliminate membership queries completely. The only other use of MQs in algorithm Learn-MQ is in Pair-MQ, where membership queries are used to discover if the pairing generated is still positive.

Instead of using a new query type to discover such pairings (such an approach is used in algorithm Learn-PQ), we instead apply a simple restriction on the equivalence oracle. Specifically, we require that the EQ always return a negative counter-example if one exists. Such an oracle is called *negatively-biased.*

Definition 4. *An oracle that answers equivalence queries is called* negatively-biased *if it always returns a negative counter-example for H when one exists.*

With this restriction we can guarantee that, upon a positive counter-example, our hypothesis H is guaranteed to be correct (albeit under-specified). Instead of searching for pairings using membership queries, we will search for the correct pairing by testing each one out in the hypothesis. If the pairing is incorrect, we will get a negative counter example that the new clause incorrectly covers. If we get a positive example, then the pairing must be correct.

Algorithm Learn-ROQ demonstrates how such an approach would work. A positive counter-example is minimized using Min-ROQ as we saw before. Then, the algorithm steps through every element in S, testing all possible pairings. A pairing is tested by directly adding it to the hypothesis and asking an equivalence query. If a negative counter-example is returned, the search continues. Otherwise, the algorithm replaces S with S' (the candidate for S) and starts over with the most recent positive example. If all pairings are eliminated via negative counter-examples then the example is appended to S as a new clause. A new positive example is received and the algorithm continues as before.

```
1  $S = \emptyset$, $H = \emptyset$, $E = \mathbf{EQ}(H)$
2  repeat
3      if $E = done$ then return $H$
4      $E' = \text{Min-ROQ}(E, \emptyset)$
5      for each $s \in S$ do
6          for every pairing $J$ between $s$ and $E'$ do
7              $S' = S \cup \{J\}$
8              $H = \mathbf{variabilize}(S')$
9              $E = \mathbf{EQ}(H)$
10             if $E$ is a positive counter example then
11                 $S = S'$
12                 Goto line 3
13     $S = S \cup \{E'\}$
14     $H = \mathbf{variabilize}(S)$
15     $E = \mathbf{EQ}(H)$
```

Algorithm Learn-ROQ. Learns $\mathcal{H}_D(P)$ using EQ and ROQ queries

Theorem 2. *The number of equivalence queries in algorithm Learn-ROQ is bound by $|P|mk^a(1 + mk^k)$. No membership queries are used. The number of relevant object queries is bound by $|P|mk^a$.*

Proof. (sketch) As before, the algorithm removes at most one literal or adds a new clause on every positive example. The number of positive examples is bound by $|P|mk^a$. Each example requires one ROQ, yielding the same bounds.

If a negative example is returned from the EQ oracle, then a pairing was incorrect and the algorithm tries to find a new one. There are mk^k total pairings for a single example, so each positive example requires at most mk^k negative examples to test pairings. The total number of examples is $|P|mk^a(1 + mk^k)$, which is equal to the number of EQs. □

While Learn-ROQ no longer uses membership queries, the number of equivalence queries has greatly increased. This is beneficial in situations where examples are plentiful but asking for new labelings is costly. A great deal of the examples ($|P|m^2k^{(a+k)}$ at most) provided by the equivalence query are negative examples, implying that this algorithm might be a good match for domains where negative examples are easy to provide.

Finally, Learn-ROQ provides motivation for an algorithm that successfully learns with mislabeled negatives. Often many real-world learning problems have a strong world model (i.e. creating an example by randomly inserting literals into a clause is very unlikely to produce a positive example). Unfortunately, the problem of a single mislabeled example is difficult to recover from with the framework in this paper, as there is no mechanism for noise. We believe a solution would be very useful in practice, and hope future work builds on this approach.

5.3 Imperfect Relevance Oracles

In all likelihood, a human answering ROQs would be unlikely to behave perfectly. This could be due to human error or in some cases due to interface issues. For example, in some domains there may be classes of objects for which the human is able to interact with as well as other objects that are internal to the domain encoding and hence are inaccessible. In such cases, imperfect heuristics might be used to determine relevance for the second class of objects.

Here we consider learning in the presence of specific types of imperfect ROQ oracle that are either verbose (includes objects that are not relevant) or conservative (does not include all the relevant objects). We will require the MQ and EQ oracles to operate perfectly. This restriction is not unreasonable as different query types may have different costs or underlying mechanisms associated with them.

Verbose Oracles. Recall that for an example E covered by multiple target clauses there are multiple possible ROQ answers, depending which covering clause C the oracle considers. When an oracle answers based on a clause C, we say that its answer is relative to C.

Definition 5. *An ROQ oracle is* (j, f)-verbose *if for at most j clauses in the target theory T, the oracle's answer Q' relative to any of those j clauses is a superset of the true set of relevant objects with at most f additional objects.*

Note that we can directly use a (j, f)-verbose oracle to answer ROQs in Learn-MQ-ROQ and still guarantee correct learning. However, the queries required increases with the amount of verbosity as quantified by j and f.

Theorem 3. *For any $T \in \mathcal{H}_D(P)$, Learn-MQ-ROQ, using a (j, f)-verbose ROQ oracle, learns an equivalent hypothesis. Compared to a perfect ROQ oracle, the number of EQs and ROQs increases by at most $|P|j(k+f)^a$ and the total number of MQs is now bound by $(|P|j(k+f)^a+|P|(m-j)k^a)(j(k+f)^{(k+f)}+(m-j)k^k)$.*

Proof. (sketch) If we have an extra f objects returned from a ROQ for j clauses in T, then the size of an example after minimization is bounded by $|P|(k+f)^a$. On the other $m-j$ clauses, the size remains the same as before $(|P|k^a)$. Therefore, if each example removes one literal, the total number of examples (and hence EQs) is bound by $|P|j(k+f)^a + |P|(m-j)k^a$. This is the same bounds for ROQs.

As before, the worst case for membership queries is when all possible pairings must be searched. The total number of pairings for an example is bound by $(m-j)k^k$ for clauses in S that the oracle does not make a mistake on, and $j(k+f)^{(k+f)}$ for the clauses that it does. Adding this up and multiplying by the number of examples gives the bounds $(|P|j(k+f)^a + |P|(m-j)k^a)(j(k+f)^{(k+f)} + (m-j)k^k)$. □

Unfortunately, it appears that for domains where n (max objects) is not exceedingly large, directly using Learn-MQ-ROQ with a (j, f)-verbose oracle can result in many more membership queries compared to the base learning algorithm Learn-MQ. It is an open question about whether it is possible to improve the dependence on j and f.

Conservative Oracles. Next, we restrict the oracle in the opposite direction where relevant objects can be missing.

Definition 6. *An ROQ oracle is* (j, f)-conservative *if, for at most j clauses in the target theory T, the oracle's answer Q' relative to any of those j clauses is a subset of the true set of relevant objects with no more than f missing relevant objects.*

Because relevant objects are missing, we cannot directly use Learn-MQ-ROQ or Learn-ROQ as our examples may be overly general. This is because conservative answers lead to overly general minimized examples, which break the assumptions of the merging processes. Instead we give a new algorithm Learn-MQ-ROQ-Conservative that is similar to the base algorithm. An important change is that the pairing algorithm Pair-MQ is slightly modified so that it only considers pairing together examples that contain the same number of objects. This new modified algorithm is referred to as *Pair-MQ-Con*.

Upon a positive example, Learn-MQ-ROQ-Conservative minimizes it using the ROQ oracle and then searches for pairings. If no pairing was found between E' (the minimized example) and the set of positive examples S, the algorithm checks to see if E' is truly a positive example using an MQ. If it is not positive then a relevant object was missed (the oracle made a mistake). The mistake is resolved by re-minimizing the example using Min-MQ. Since all objects given by the conservative ROQ are relevant, they are passed to Min-MQ so the routine does not spend time testing them. After minimization the correct example is now passed to Pair-MQ-Con for pairing. Using Pair-MQ-Con in place of Pair-MQ implies that the algorithm will never try to pair E' to the same example s more than once. If a relevant object is missing and E' does not pair with anything in S, then the number of objects in E' will necessarily increase (as the membership query will detect it and employ Min-MQ. Because the size has increased, E' will not be checked against previously checked elements in S and avoid a significant amount of redundant work.

```
$S = \emptyset$, $H = \emptyset$
repeat
    $E = EQ(H)$
    if $E = done$ then return $H$
    if $E$ is a positive counter example then
        $E' =$ Min-ROQ$(E)$
        $S =$ Pair-MQ-Con$(E', S)$
        if $E' \in S$ ($E'$ did not pair with any elements) and MQ$(E')$ is false then
            $S = S - \{E'\}$
            $E' =$ Min-MQ$(E,$ objects$(E'))$
            $S =$ Pair-MQ-Con$(E', S)$
    $H =$ variabilize$(S)$.
```

Algorithm Learn-MQ-ROQ-Conservative. Learns $\mathcal{H}_D(P)$ using EQ, MQ, and conservative ROQ queries

Theorem 4. *For any $T \in \mathcal{H}_D(P)$, Learn-MQ-ROQ-Conservative, using a (j, f)-conservative ROQ oracle, learns an equivalent hypothesis. Compared to Learn-MQ-ROQ with a perfect oracle the maximum number of EQs and ROQs are unchanged and the total number of MQs is bound by $|P|jk^a(mk^k + 1 + n - (k - f)) + |P|m(m - j)k^{(a+k)}$.*

Proof. (sketch) The number of examples remains the same as Learn-MQ-ROQ ($|P|mk^a$), giving the bounds on equivalence queries and relevant object queries. If the relevant object oracle makes a mistake ($j > 0$), then the algorithm searches for pairings. The worst case occurs when all clauses in the target hypothesis have the same number of variables, so this search for pairings adds no membership queries as E' has $k - f$ variables.

Before generating H, the algorithm asks a membership query to verify the example is actually positive (which it is not). The search for the true number of relevant objects in an example takes $n - (k - f)$ membership queries, as that is how many objects there are that might possibly be relevant. Finally, the search for pairings is restarted, giving mk^k total pairings that can be found.

When an example does not have a mistake, the number of membership queries remains the same as the original algorithm (mk^k for finding the correct pairing). Multiplying the number of MQs for each example by the size of each type of example gives $|P|jk^a(mk^k+1+n-(k-f))+|P|m(m-j)k^{(a+k)}$, which is what is provided above. □

Algorithm Learn-MQ-ROQ-Conservative has a much more reasonable increase of membership queries with respect to j and f compared to the verbose setting. Instead of an exponential increase of MQs with f, the change is now linear. It remains an open problem as to whether it is possible to remove membership queries altogether when using a conservative ROQ oracle.

Verbose vs. Conservative. The above results show that clearly, when MQs are available, a conservative oracle is superior to a verbose oracle with respect to the number of MQs. This is because it is much easier to identify an example missing relevant objects than an example with extra objects. This has implications for domains where heuristics are used to judge relevance of objects that are not available via an interface for a human teacher to judge relevance. In particular, it suggests that when MQs are available using a conservative heuristic is desirable.

5.4 Pairing Queries

The majority of the MQs in Learn-MQ are used in the search over pairings. However, it is plausible that often a teacher will be able to directly indicate which objects between two existing examples "correspond" with respect to the target concept. For this purpose we introduce pairing queries.

Definition 7. *A* pairing query (PQ) *is a query that, given two positive examples E_1 and E_2, returns* **false** *if there is no clause $C \in T$ that covers both E_1 and E_2. Otherwise it picks a target clause C that covers both examples via substitutions θ_1 and θ_2 and returns a 1-to-1 mapping between objects in θ_1 and objects in θ_2 where objects are mapped together if they correspond to the same variable in θ_1 and θ_2.*

Note that by definition the mapping returned by a PQ will only involve relevant objects. In this sense, PQs are strictly more powerful than ROQs. Pairing queries can be directly placed in Learn-MQ-ROQ by replacing the loop that searches over pairings. The algorithm directly asks if two examples pair together and asks for a mapping P. If the query is false, the algorithm proceeds through the set S, appending the example if no pairing is found. Otherwise, a pairing is calculated using the mapping P and the algorithm proceeds normally. This is demonstrated in algorithm Learn-PQ. From previous results we can easily bound the required number of EQs and PQs.

Proposition 2. *For any $T \in \mathcal{H}_D(P)$, Learn-PQ, learns an equivalent hypothesis using no more than $|P|mk^a$ EQs and $|P|m^2k^a$ PQs. No MQs or ROQs are required.*

It is unclear how much more difficult it would be for PQs to be answered compared to ROQs; ultimately experience with real users is needed. It is not difficult to imagine a natural interface for PQs in many domains. A system could be implemented that shows the two examples to a user (e.g. two chess boards) and asks for a matching between objects that have the same role in each example. This is similar to asking "What do these two examples have in common?" which is a question that seems natural in a learning environment. The problem of learning with noisy pairing queries is currently open and an important future direction.

```
$S = \emptyset$, $H = \emptyset$
repeat
    $E = EQ(H)$
    if $E = done$ then return $H$
    if $E$ is a positive counter example then
        for each $s \in S$ do
            $P = \mathbf{PQ}(E, S_i)$
            if $P$ is not false then
                Let $J$ be the pairing generated using mapping $P$ with $E$ and $S_i$, where objects not found in $P$ are removed by dropping literals that reference them.
                $S = S \cup \{J\} - s$
        if no pairings were found then
            $S = S \cup \{E\}$
    $H = \mathbf{variabilize}(S)$
```

Algorithm Learn-PQ. Learns $\mathcal{H}_D(P)$ using EQ and PQ queries

6 Summary and Future Work

We have shown algorithms that allow for object-based queries to learn in a polynomially-bounded number of queries. We specifically focused on relevant object queries and pairing queries, but these are by no means the only object-based queries that may be useful. It is our hope that the results shown above motivate a further examination into new query types.

While it was claimed that object-based queries might be easier for a human to use than traditional membership queries, no user study has been performed to evaluate this claim. This motivates a user study where users act as various oracle types to try to teach a concept. Users could also be asked what sort of information they would like to be able to express, possibly giving motivation for a new query type or annotation.

Finally, we examined two types of error-bound object-based queries. Both types were deterministic, but such a restriction may not be practical. A probabilistic model of oracle responses may be more likely in practice. Other error models can also be considered, such as a mixing of the two error types, or providing a large set of negative examples that are correct with some probability.

Acknowledgements. The authors acknowledge support of the Defense Advanced Research Projects Agency (DARPA) under grant HR0011-07-C-0060. 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 DARPA.

References

1. Angluin, D.: Queries and concept learning. Machine Learning 2, 319–342 (1988), http://dx.doi.org/10.1023/A:1022821128753, doi:10.1023/A:1022821128753
2. Angluin, D., Frazier, M., Pitt, L.: Learning conjunctions of horn clauses. Machine Learning 9, 147–164 (1992), http://dx.doi.org/10.1007/BF00992675
3. Arias, M., Khardon, R., Maloberti, J.: Learning horn expressions with LOGAN-H. The Journal of Machine Learning Research 8, 549–587 (2007)
4. Baum, E.B., Lang, K.: Query learning can work poorly when a human oracle is used. In: IJCNN (1992)
5. Feldman, V., Shah, S.: Separating models of learning with faulty teachers. Theor. Comput. Sci. 410, 1903–1912 (2009), http://portal.acm.org/citation.cfm?id=1519541.1519713
6. Khardon, R.: Learning function-free horn expressions. Machine Learning 37, 241–275 (1999), http://dx.doi.org/10.1023/A:1007610422992
7. McDaniel, R.G., Myers, B.A.: Getting more out of programming-by-demonstration. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI is the Limit, CHI 1999, pp. 442–449. ACM, New York (1999), http://doi.acm.org/10.1145/302979.303127
8. Plotkin, G.D.: A Note on Inductive Generalization. Machine Intelligence 5, 153–163 (1970)
9. Raedt, L.D., Dzeroski, S.: First-order jk-clausal theories are pac-learnable. Artificial Intelligence 70(1-2), 375–392 (1994), http://www.sciencedirect.com/science/article/B6TYF-47WT97K-F/2/55a8bf6608387d13b5bfa7a55e4b50d2
10. Reddy, C., Tadepalli, P.: Learning first-order acyclic horn programs from entailment. In: Page, D. (ed.) ILP 1998. LNCS, vol. 1446, pp. 23–37. Springer, Heidelberg (1998), http://dx.doi.org/10.1007/BFb0027308
11. Reddy, C., Tadepalli, P.: Learning horn definitions: Theory and an application to planning. New Generation Computing 17, 77–98 (1999), http://dx.doi.org/10.1007/BF03037583, doi:10.1007/BF03037583
12. Valiant, L.G.: A theory of the learnable. Commun. ACM 27, 1134–1142 (1984), http://doi.acm.org/10.1145/1968.1972
13. Walker, T., Natarajan, S., Kunapuli, G., Shavlik, J., Page, D.: Automation of ilp setup and search via user provided relevance and type information. In: Inductive Logic Programming (2010)

Fast Support Vector Machines for Structural Kernels

Aliaksei Severyn and Alessandro Moschitti

Department of Computer Science and Engineering,
University of Trento,
Via Sommarive 5, 38123 POVO (TN) - Italy
{severyn,moschitti}@disi.unitn.it

Abstract. In this paper, we propose three important enhancements of the approximate cutting plane algorithm (CPA) to train Support Vector Machines with structural kernels: (i) we exploit a compact yet exact representation of cutting plane models using directed acyclic graphs to speed up both training and classification, (ii) we provide a parallel implementation, which makes the training scale almost linearly with the number of CPUs, and (iii) we propose an alternative sampling strategy to handle class-imbalanced problem and show that theoretical convergence bounds are preserved. The experimental evaluations on three diverse datasets demonstrate the soundness of our approach and the possibility to carry out fast learning and classification with structural kernels.

1 Introduction

Various kernels have been successfully applied to different Natural Language Processing (NLP) tasks, e.g. [13,5,17,12,6,2]. However, previous work is limited to relatively small datasets. Indeed, the major drawback of kernel methods is the necessity to carry out learning in dual spaces, where training complexity typically is quadratic in the number of instances.

Recently, a number of efficient CPA-based algorithms have been proposed [10,8]. Unfortunately, these algorithms scale well only when linear kernels are used. To address slow learning with non-linear kernels [12] propose to extract basis vectors to compactly represent cutting plane models, which speeds up both classification and learning. However, this requires to solve a non-trivial optimization problem when arbitrary kernel functions are used. Finding a set of basis vectors in high-dimensional spaces produced by arbitrary kernels, structural kernels in particular, is an open research area and is definitely worth further exploration.

Another approach of adapting CPA for non-linear kernels by reducing the number of kernel evaluations is studied in [23], where sampling is used to reduce the number of basis functions in the kernel expansion. [16] showed that the same algorithm can be successfully applied to SVM learning with structural kernels on very large data obtaining speedup up factors up to 10 over conventional SVMs.

In this paper, we provide three important improvements of the approximate CPA with sampling. The proposed techniques make SVMs with structural kernels

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 175–190, 2011.

a viable tool to tackle real-world tasks. In particular, we first present an idea to use (Directed Acyclic Graphs) DAGs to compactly represent cutting plane models computed at each iteration of the CPA algorithm. This has the benefit of reducing the number of expensive kernel evaluations, since DAGs provide the means to avoid redundant computations over shared substructures. We present two algorithms that deliver impressive speedups for both training and testing. We also parallelize the code improving scalability even further. Finally, we provide an effective and sound method to handle class imbalanced datasets, which plays an important role to obtain the optimal balance between Precision and Recall.

2 Preliminaries: Cutting Plane Algorithm with Sampling

In this section, we illustrate a re-elaborated version of the cutting plane method (originally proposed in the context of structural SVMs) for binary classification. After briefly explaining it for linear SVMs, we point out the main source of inefficiency for the case when kernels are used. Next we present the idea of [23] to use sampling to alleviate high training costs for SVMs with non-linear kernels.

2.1 Cutting-Plane Algorithm (Primal)

Consider a slight modification of SVM training problem, known as a 1-slack reformulation [10], to derive CPA for binary classification[1]:

$$\begin{aligned} \underset{\boldsymbol{w},\xi\geq 0}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\xi \\ \text{subject to} \quad & \frac{1}{n}\sum_{i=1}^{n} c_i y_i \boldsymbol{w}\cdot\boldsymbol{x}_i \geq \frac{1}{n}\sum_{i=1}^{n} c_i - \xi, \ \forall \boldsymbol{c}\in\{0,1\}^n \end{aligned} \tag{1}$$

where binary vector $\boldsymbol{c} = (c_1, \ldots, c_n) \in \{0,1\}^n$ forms a constraint that is a linear combinations of the constraints $y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i) \geq 1 - \xi_i$.

The CPA is presented in Alg. 1. It starts with an empty set of constraints S and computes the optimal solution to the unconstrained problem (1). Next, the algorithm forms a binary vector $\boldsymbol{c}$ to compute a cutting plane model defined by its offset $d^{(t)} = \frac{1}{n}\sum_{i=1}^{n} c_i$ and gradient $\boldsymbol{g}^{(t)} = \frac{1}{n}\sum_{i=1}^{n} c_i y_i x_i$ (lines 5-9). This produces a constraint $\boldsymbol{w}\cdot\boldsymbol{g}^{(t)} \geq d^{(t)} - \xi$ that is violated the most by the current solution $\boldsymbol{w}$, which is included in the set of active constraints S (line 10). This way, a series of successively tightening approximations to the original problem is constructed. The algorithm stops when no constraints are violated by more than ϵ, which is formalized by the criteria in line 12.

2.2 Cutting-Plane Algorithm (Dual)

To enable the use of kernels, we need to solve the Wolfe dual of the problem (1). Its solution $\boldsymbol{w}$ lies in the feature space defined by a kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_k) = \phi(\boldsymbol{x}_i)\cdot$

[1] Here we fix the bias term b at zero, as it could be easily incorporated in feature vectors as an additional constant.

Algorithm 1. Cutting Plane Algorithm (primal)

```
 1: Input: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, $C$, $\epsilon$
 2: $S \leftarrow \emptyset; t = 0$
 3: repeat
 4:     $(\boldsymbol{w}, \xi) \leftarrow$ optimize (1) over the constraints in $S$
        /* find a cutting plane */
 5:     for $i = 1$ to $n$ do
 6:         $c_i^{(t)} \leftarrow \begin{cases} 1 \textbf{ if } \big(y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i) \leq 1\big) \\ 0 \textbf{ otherwise} \end{cases}$
 7:     end for
 8:     $d^{(t)} \leftarrow \frac{1}{n} \sum_{i=1}^{n} c_i$
 9:     $\boldsymbol{g}^{(t)} \leftarrow \frac{1}{n} \sum_{i=1}^{n} c_i y_i \phi(\boldsymbol{x_i})$
        /* add a constraint to the set of constraints */
10:     $S \leftarrow S \cup \{(d^{(t)}, \boldsymbol{g}^{(t)})\}$
11:     $t \leftarrow t + 1$
12: until $\boldsymbol{w} \cdot \boldsymbol{g}^{(t)} \geq d^{(t)} - \xi + \epsilon$
13: return $\boldsymbol{w}, \xi$
```

$\phi(\boldsymbol{x}_i)$. It can be easily verified (by deriving the the dual from (1)) that primal and dual variables are connected via:

$$\boldsymbol{w} = \sum_{j=1}^{t} \alpha_j \boldsymbol{g}^{(j)}, \tag{2}$$

where $\boldsymbol{g}^{(j)} = \frac{1}{n} \sum_{k=1}^{n} c_k^{(j)} y_k \phi(\boldsymbol{x}_k)$ denotes the gradient of the cutting plane model added at iteration j and t is the size of the set S.

As one can see, with the use of kernels the gradient $\boldsymbol{g}^{(j)}$ (that also defines the most violated constraint (MVC) added at iteration j) cannot be compactly represented as in the linear case by simply summing up n feature vectors since it now lies in the feature space spanned by $\phi(\cdot)$. We will address the problem of compact representation of the cutting plane models in the next section.

Computing an inner product between the weight vector $\boldsymbol{w}$ and an example $\boldsymbol{x}_i$ involves the sum of kernel evaluations for each example $\boldsymbol{x}_k$ in the constraint j for each constraint in S. In particular, using the expansion of $\boldsymbol{w}$ from (2), the inner product required to compute the MVC (steps 5-9 in the Alg. 1), renders as:

$$\boldsymbol{w} \cdot \phi(\boldsymbol{x}_i) = \sum_{j=1}^{t} \alpha_j \boldsymbol{g}^{(j)} \cdot \phi(\boldsymbol{x}_i) = \sum_{k=1}^{n} \Big(\sum_{j=1}^{t} \frac{1}{n} \alpha_j c_k^{(j)} y_k \Big) K(\boldsymbol{x}_i, \boldsymbol{x}_k), \tag{3}$$

The analysis of the inner product given by (3) reveals that the number of kernel evaluations at each iteration is $O(tn^2)$. Indeed, the number of non-zero elements in each $\boldsymbol{g}^{(j)}$ is proportional to the number of support vectors which grows linearly with the training size n [19]. Summing over all constraints in the set S, the complexity of (3) is $O(tn)$. Since the inner product (3) needs to be computed for each training example (lines 5-7 in Alg. 1) we obtain $O(tn^2)$ scaling behavior at each iteration.

The obtained quadratic scaling in the number of examples makes cutting plane training for non-linear SVMs prohibitively expensive for even medium-sized datasets. To address this limitation [23] proposed to construct approximate cuts by sampling r examples from the training set. The idea is to replace the expensive computation of the MVC (lines 5-7, Alg. 1) over all training examples n by a sum over a smaller sample r, s.t. the number of examples in $\boldsymbol{g}^{(j)}$ is reduced from $O(n)$ to $O(r)$. In this case the double sum of kernel evaluations in (3) reduces from $\sum_{i,j=1}^{n} K(\boldsymbol{x}_i, \boldsymbol{x_j})$ to a more tractable $\sum_{i,j=1}^{r} K(\boldsymbol{x}_i, \boldsymbol{x_j})$.

3 Fast CPA for Structural Kernels

In this section we present an approach to significantly speed up the approximate CPA for structural kernels. We observe that for convolution structural kernels, e.g. tree kernels, the cutting plane model can be compactly represented as a Directed Acyclic Graph (DAG). This helps to speed up both the training and classification as the repeating kernel evaluations over shared substructures can be avoided. Most interestingly this approach can be parallelized during training thus making structural kernel learning practical on larger datasets.

3.1 Compacting Cutting Plane Models Using DAGs

In the previous section we have seen that computing an MVC at each iteration involves quadratic number of kernel evaluations. Using smaller samples to approximate the cutting plane helps to reduce the number of kernel evaluations.

Here we explore another avenue to reduce the number of kernel computations when convolution structural kernels are used. Indeed, when applied to structural data such as sequences, trees or graphs, we can take advantage of the fact that many examples share common sub-structures. Hence, we can use a compact representation of a cutting plane model to avoid redundant computations over repeating sub-structures. In particular, when dealing with tree-structured data, a collection of trees can be compactly represented as a DAG [1]. In the following we briefly introduce the idea behind using DAGs to compactly represent a tree forest and then show how it applies to speed up the learning algorithm.

3.2 DAG Tree Kernels

A DAG can efficiently represent a set of trees by including only the unique subtrees and accounting for the frequency of the repeated substructures. Fig. 1 shows three syntactic trees on the left and the resulting DAG on the right. As we can see, the subtree of the noun phrase `[NP [D a][N car]]` is repeated in two trees, thus the frequency of the corresponding node is updated to 2. Also smallesubtrees such as `[D a]` and `[D car]` are shared with a frequency of 3. The two subtrees rooted in `VP` are different and require different roots but they can still share some of their subparts, e.g. `[V buy]`.

Given a collection of trees, there are various methods to efficiently build a corresponding DAG and allow for fast access to its tree nodes, see for example [1].

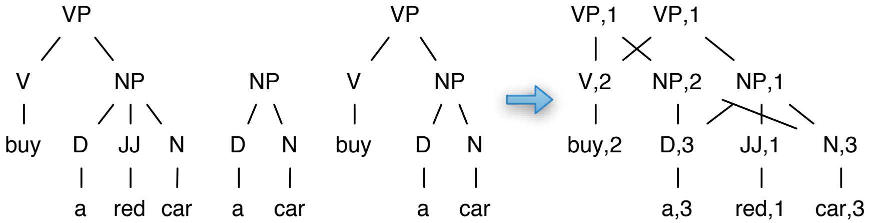


Fig. 1. Three syntactic trees and the resulting DAG

In our approach, for each node in a tree, we generate a string representation of its subtree. This requires linear time in the number of tree nodes and can be done at the preprocessing step. These strings are unique identifiers of each respective node and serve as keys in the hash table, whose values are pointers to the corresponding nodes. To perform efficient search within a DAG, we maintain a simple and efficient nested structure of two associative arrays. The first is a hash table, which given a node retrieves the set of nodes associate with the same production rule. Each entry in the retrieved set contains a tuple of a pointer to the node and its current frequency. In this way we can efficiently enumerate all the candidate substructures to compute the tree kernel [4] between a DAG and a given tree.

Tree Kernels (TKs). Convolution TKs compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. For this purpose, let the set $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$ be the substructure space and $\chi_i(n)$ be an indicator function, equal to 1 if the target t_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over T_1 and T_2 is $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{T}|} \chi_i(n_1)\chi_i(n_2)$. The latter is equal to the number of common fragments rooted in the n_1 and n_2 nodes.

Theorem 1. *Let D be a DAG representing a tree forest F and $K_{dag}(D, T_2) = \sum_{n_1 \in N_D} \sum_{n_2 \in N_{T_2}} f(n_1)\Delta(n_1, n_2)$ then*

$$\sum_{T1 \in F} TK(T_1, T_2) = K_{dag}(D, T_2), \tag{4}$$

where $f(n_1)$ is the frequency associated with n_1 in the DAG.

Proof. Let $\mathcal{S}(F)$ the set of possible subtrees of F, i.e. the substructures whose leaves coincide with those of the original tree (in general $\mathcal{T} \neq \mathcal{S}$), then $\sum_{T_1 \in F} TK(T_1, T_2) = \sum_{T_1 \in F} \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) = \sum_{T_1 \in F} \sum_{\substack{n_1 : t \in \mathcal{S}(T_1) \\ n_1 = r(t)}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where $r(t)$ is the root of the subtree t. The last expression is equal to $\sum_{\substack{n_1 : t \in \mathcal{S}(F) \\ n_1 = r(t)}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$. Let $\mathcal{S}'$ the unique subtrees of

$\mathcal{S}$, we can rewrite the above equation as: $\sum_{\substack{n_1:t\in\mathcal{S}'(F)\\ n_1=r(t)}} f(n_1)\sum_{n_2\in N_{T_2}} \Delta(n_1,n_2)=$ $\sum_{\substack{n_1:t\in D\\ n_1=r(t)}} \sum_{n_2\in N_{T_2}} f(n_1)\Delta(n_1,n_2)= \sum_{n_1\in D}\sum_{n_2\in N_{T_2}} f(n_1)\Delta(n_1,n_2)$. □

3.3 Fast Computation of the MVC on Structural Data

Having introduced the DAG tree kernel, we redefine the inner product (3) required to compute the MVC by compacting $\boldsymbol{g}^{(j)}$ into a single DAG model $\boldsymbol{dag}^{(j)}$:

$$\boldsymbol{w}\cdot\phi(\boldsymbol{x}_i)=\sum_{j=1}^{t}\alpha_j K_{dag}(\boldsymbol{dag}_{(j)},\boldsymbol{x}_i) \tag{5}$$

Unlike (3), where each cutting plane $\boldsymbol{g}^{(j)}$ is an arithmetic sum of training examples, here we take advantage of the fact that a collection of trees can be efficiently put into an equivalent DAG. Please note that computing a kernel $K_{dag}(\cdot,\cdot)$ between an example and a DAG that represents a collection of trees yields an exact kernel value as shown in Th. 1. The benefit of such representation comes from the efficiency gains obtained by speeding up kernel evaluations over the sum of examples compacted into a single DAG.

Now we are ready to present the new cutting plane algorithm (see Alg. 2) adapted for the use of structural kernels. The weight vector $\boldsymbol{w}$ (line 6) is now expanded over the dual variables α_j obtained by solving Wolfe dual of (1) (line 5) and cutting plane models, each compactly represented by $\boldsymbol{dag}^{(j)}$. To compute the MVC we use a smaller set of examples uniformly sampled from the original training set. A binary vector $\boldsymbol{c}$ formed in (lines 8-12) defines the examples that are inserted into a DAG model (line 11). The obtained algorithm preserves all the theoretical benefits of the approximate CPA with sampling, while reducing the number of expensive kernel evaluations to compute the MVC.

To benefit even more from the compact representation offered by DAGs, we can put all cutting planes from the set S into a single DAG, such that the inner product (3) is reduced to a single kernel evaluation:

$$\boldsymbol{w}\cdot\phi(\boldsymbol{x}_i)=K_{dag}(\widehat{\boldsymbol{dag}}_{(t)},\boldsymbol{x}_i) \tag{6}$$

where $\widehat{\boldsymbol{dag}}_{(t)}$ at iteration t is built by inserting nodes from $\boldsymbol{dag}_{(j)}$ together with the frequency counts multiplied by the value of the corresponding dual variable α_j. This ensures that a single K_{dag} evaluation over the full DAG model makes Eq. 6 equivalent to computing a weighted sum of K_{dag} using individual $\boldsymbol{dag}_{(j)}$ in Eq. 5. Even though $\widehat{\boldsymbol{dag}}_{(t)}$ has to be re-built at each iteration to accommodate updated vector $\boldsymbol{\alpha}$, this imposes little computational overhead in practice. Another computational drawback of using full DAG model compared to the set of $\boldsymbol{dag}_{(j)}$ is that in the former case we need to compute the update of the Gram matrix column (line 4 in Alg.2) $G_{it}=\boldsymbol{g}^{(i)}\cdot\boldsymbol{g}^{(t)}$ for $1\le i\le t$, while in the latter case it is obtained automatically from computing Eq. 5.

Even though the worst-case complexity for computing the MVC using both variants of using DAGs is still $O(r^2)$, it is highly unlikely to observe in practice,

Algorithm 2. Cutting Plane Algorithm (dual) using DAG model representation

```
1: Input: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, $C$, $\epsilon$
2: $S \leftarrow \emptyset$; $\boldsymbol{dag} \leftarrow 0$; $t = 0$;
3: repeat
4:    Update the Gram matrix $G$ with a new constraint
5:    $\boldsymbol{\alpha} \leftarrow \text{argmax}_{\boldsymbol{\alpha} \geq 0} \boldsymbol{h}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T G \boldsymbol{\alpha}$, s.t. $\boldsymbol{\alpha}^T \mathbf{1} \leq C$ where $h_i = d^{(i)}$ and $G = \boldsymbol{g}^{(i)} \cdot \boldsymbol{g}^{(j)}$
6:    $\boldsymbol{w} = \sum_{j=1}^{|t|} \alpha_j \boldsymbol{dag}^{(j)}$
7:    Sample $r$ examples from the training set
      /* find a cutting plane */
8:    for $i = 1$ to $r$ do
9:       $c_i^{(t)} \leftarrow \begin{cases} 1 & \textbf{if } \left(y_i \sum_{j=1}^{t} \alpha_j K_{dag}(\boldsymbol{dag}_j, \boldsymbol{x}_i) \leq 1\right) \\ 0 & \textbf{otherwise} \end{cases}$
10:   end for
11:   $\boldsymbol{dag}^{(t)} = \text{build\_dag}(\boldsymbol{c})$
12:   $d^{(t)} = \frac{1}{r} \sum_{i=1}^{r} c_i$
13:   /* add dag to the active set */
14:   $S \leftarrow S \cup \{(d^t, \boldsymbol{dag}^t)\}$
15:   $t = t + 1$
16: until $d^{(t)} - \boldsymbol{w} \cdot \boldsymbol{dag}^{(t)} \leq \xi + \epsilon$
17: return $\boldsymbol{w}, \xi$
```

where input examples tend to share many common substructures. This speeds up both training and classification by avoiding redundant kernel computations.

3.4 Parallelization

The modular nature of the CPA suggests easy parallelization. In fact, in our experiments, we observed that at each iteration 95% of the total learning time is spent on computing the MVC (steps 8-12, Alg. 2). This involves computing Eq. 5 over the set of individual DAGs or Eq. 6 using full DAG model for r training examples in the sample. This observation suggests high parallelizability of the code: using p processors the complexity of this pre-dominant part can be brought down from $O(r^2)$ to $O(r^2/p)$.

4 Handling Class-Imbalanced Data

In this section, we extend the theory of cutting-plane algorithm to tackle class-imbalance problem. Our approach is based on an alternative sampling strategy that is effective for tuning up Precision and Recall on class-imbalanced data. We also provide a convergence proof of the proposed algorithm.

4.1 Cost-Proportionate Sampling

Conventional SVM problem formulation allows for natural incorporation of example dependent importance weights into the optimization problem. We can modify the objective function to include example dependent cost factors:

$$\begin{aligned} \underset{w,\xi_i \geq 0}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_i^n z_i \xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{x_i}) \geq 1 - \xi_i,\ 1 \leq i \leq n \end{aligned} \tag{7}$$

where z_i is the importance weight of example i and $\frac{1}{n}\sum_i^n z_i \xi_i$ serves as an upper bound on the total cost-sensitive empirical risk. This problem formulation where there is an individual slack variable ξ_i for each example is typically referred to as "n-slack" formulation.

In the dual space, the example-dependent costs captured by cost factors z_i translate into the box constraints imposed on each dual variables: $0 \leq \alpha_i \leq z_i C$, $1 \leq i \leq n$ such that the $z_i C$ sets an upper bound on the values of α_i. This feature to integrate importance weights z_i in the problem formulation is implemented in SVM-light software.

This natural modification of the quadratic problem, is, however, difficult to incorporate in the case of 1-slack formulation (1). Indeed, in the case of 1-slack formulation we have a single slack variable ξ that is shared among all the constraints. More importantly, moving to the dual space, the box constraints $0 \leq \alpha_i \leq C$ are no longer for each individual dual variable but for a sum: $\sum_i \alpha_i$. This makes the 1-slack problem formulation difficult to incorporate importance weights directly. Nevertheless, the idea of approximating the cutting plane model at each iteration via sampling suggests a straightforward solution.

Indeed, we can extend the original CPA to the case of cost-sensitive classification. A straight-forward way to do this is instead of using uniform sampling to build an approximation to the cutting plane model at each iteration (steps 8-12 in Alg. 2), we can draw examples according to their importance weights using the cost-proportionate rejection sampling technique (Alg. 3).

Algorithm 3. Cost-proportionate rejection sampling

```
Pick example $(\boldsymbol{x_i}, y_i, z_i)$ at random
Flip a coin with bias $z_i/Z$
if heads then
    keep the example
else
    discard it
end if
```

Here z_i is the importance weight of the i-th example and Z is an upper bound on any importance value in the dataset. This process is repeated until we sample the required number of examples r. This modification enables the control over the proportion of examples from different classes that will form a sample used to compute the MVC.

Unlike the conventional approaches for addressing the class-imbalance problem, that either under-sample the majority class or over-sample the minority class from the training data, the rejection sampling coupled with CPA does not completely discard examples from the training set. At each iteration it forms a sample according to the pre-assigned importance weights for each example, such that examples from both the majority and minority classes enter the sample in the desired proportion. This process is repeated until the algorithm converges. Thus, the learner has the chance to incorporate relevant information present in the data over a number of iterations before it converges. This way, the method preserves the global view on the dataset and no relevant information is lost during the iterative optimization process unlike in the "one-shot" sampling methods.

Another benefit of this approach is that by increasing the importance weight of the minority class, we give its examples more chance to end up in the MVC and hence, become support vectors. This way the imbalanced support-vector ratio is automatically tuned to include more examples from the minority class, which gives more control over the class-imbalance problem. Proving this property could be an interesting theoretical result.

4.2 Theoretical Analysis of the Algorithm

Cost proportionate rejection sampling allows for natural extension of the binary classification to importance weighted binary classification. It achieves this task by re-weighting the original distribution of examples D according to the importance weights of examples such that the training is effectively carried out under the new distribution $\hat{D}$.

In [24] it is shown that by transforming the original distribution D to a training set under $\hat{D}$, one can effectively train a **cost-insensitive** classifier on a dataset $\hat{D}$ such that it will minimize the expected risk under the original distribution D.

Theorem 2. (Translation Theorem; [24]) *Learning a classifier h to minimize the expected cost-sensitive risk under the original distribution D is equivalent to learning a decision function to minimize the expected **cost-insensitive** risk under the distribution $\hat{D}(x, y, z) \equiv \frac{z}{E_{(x,y,z)\sim D}[z]} D(x, y, z)$.*

The proof is a straight-forward application of the definitions and simply follows by establishing an equivalence relationship between the expected cost-sensitive risk $E_{(x,y,z)\sim D}[z\Delta(y, h(x))]$ under the original distribution D and the expected cost-insensitive risk $E_{(x,y,z)\sim \hat{D}}[\Delta(y, h(x))]$ under the transformed distribution $\hat{D}$. The theorem produces an important implication that by transforming the original distribution D to $\hat{D}$ according to example-dependent importance weights, a classifier for the cost-sensitive problem over D can be obtained with a cost-insensitive learning algorithm over D. We can use this finding to show that the convergence proof for the original CPA with uniform sampling naturally applies to the proposed version of the algorithm that uses cost-proportionate rejection sampling:

Theorem 3. (Convergence) *Assume $R = max_{1\leq i\leq n}\|\phi(\boldsymbol{x}_i)\|$, i.e. R is an upper bound on the norm of any $\phi(\boldsymbol{x}_i)$, and $\Delta = max_{1\leq i\leq n}\| \Delta(y, y_i)\|$, the number of steps required by Alg. 2 using the sampling strategy of Alg. 3 is upper bounded by $8C\Delta R^2/\epsilon^2$.*

Proof. We first note that the cost-proportionate rejection sampling (Alg. 3), used to build the approximate cutting plane model, at each step re-weights the original distribution D according to the importance weights of the examples. This means that we are effectively training a cost insensitive classifier that draws examples to build the cutting plane model from the transformed distribution $\hat{D}$. By invoking the Translation Theorem (2), we establish that, to obtain a cost-sensitive classifier that minimizes the expected risk under the original distribution D, it is sufficient to learn a cost-insensitive classifier under the transformed distribution $\hat{D}$. The CPA that draws examples from D using rejection sampling is equivalent to the original CPA applying uniform sampling to the transformed distribution $\hat{D}$. Thus, we can reutilize the proof in [23] of the convergence bounds for the original CPA with uniform sampling over $\hat{D}$. This states that CPA with uniform sampling terminates after at most $8C\mathit{\Delta}R^2/\epsilon^2$ iterations. By applying such bound, we have proved the thesis of the theorem.

Remarks. The main idea to obtain convergence bounds in [23] is to set an upper bound on the value of the dual objective and if there exists a lower bound on the minimal improvement of the dual objective at each iteration, then the algorithm will terminate in a finite number of steps.

Indeed, using the relationship between primal and dual problems, we have that a feasible solution of the primal problem (1), such as, for example: $\boldsymbol{w} = \boldsymbol{0}$, $\xi = \mathit{\Delta}$, forms an upper bound $C\mathit{\Delta}$ on the dual objective of 1. Next, in [20] it is shown that the inclusion of ϵ-violated constraint at each iteration improves the dual objective by at least $\epsilon/8R^2$. Since the dual objective is upper bounded by $C\mathit{\Delta}$, the algorithm terminates after at most $8C\mathit{\Delta}R^2/\epsilon^2$ iterations.

The derivation of the bound on the minimal improvement of the dual objective obtained at each step only depends on the values of ϵ and R and does not rely on the assumption about distribution of the examples. Also note that each cutting plane model built via rejection sampling is a valid constraint for the optimization problem (1).

5 Experiments

In our experiments we pursue a three-fold goal: (i) study the effects of compacting the cutting plane model by using DAGs on both training and classification runtimes; (ii) demonstrate the speedup factors one can obtain after straightforward parallelization offered by the CPA; and (iii) demonstrate the ability of the cost-proportionate sampling scheme to tune up Precision and Recall;

5.1 Experimental Setup

We integrated CPA with uniform sampling as described in [23] within the framework of SVM-Light-TK [14,9] to enable the use of structural kernels, e.g. tree kernels. For the DAG implementation we employ highly efficient Judy arrays[2].

[2] http://judy.sourceforge.net

For brevity, we refer to the CPA with uniform sampling as uSVM; uSVM where each cutting plane $\boldsymbol{g}^{(j)}$ is compacted into a $\boldsymbol{dag}_{(j)}$ as SDAG; uSVM with a single DAG that fits all active constraints in the set S as SDAG+; uSVM with rejection sampling as uSVM+j (Alg. 3), and SVM-light-TK as SVM. Parallel implementation relies on the OpenMP library.

To carry out learning, we used the subset tree (SST) kernel [4] since it has been indicated as the most accurate in similar tasks, e.g. [14]. As the stopping criteria of the algorithms, we fix the precision parameter ϵ at 0.001. The margin trade off parameter is fixed at 1.0. To measure the classification performance, we use Precision, Recall and F_1-score. We ran all the experiments on machines equipped with Intel® Xeon® 2.33GHz CPUs carrying 6Gb of RAM under Linux.

5.2 Data and Models

To evaluate the efficiency of the compact model representation offered by SDAG and SDAG+ algorithms with respect to uSVM, we use Semantic Role Labeling (SRL) benchmark, using PropBank annotations [15] and automatic Charniak parse trees [3]. SRL dataset has already been used to extensively test uSVM for structural kernels and we follow the same setting as described in [16].

In the next set of experiments to study the ability of uSVM+j to tune up Precision and Recall we used two different natural language datasets: TREC 10 QA[3] (training: 5,483, test: 500) and Yahoo! Answers (YA)[4](train: up to 300k, test: 10k) to perform two similar tasks of QA classification. The task for the first dataset is to select the most appropriate type of the answer from a set of given possibilities. The goal of the experiments on these relatively small datasets is to demonstrate that rejection sampling is able to effectively handle class imbalance similar to SVM. For Yahoo! Answers dataset the classification task was set up as follows. Given pairs of questions and corresponding answers learn if in a given pair the answer is the 'best' answer for a question. The goal of this experiment is to have a large classification task (300k examples in our experiments) to demonstrate that class-imbalance problem can be handled effectively at a scale where SVM becomes too slow.

5.3 Results and Analysis

Compact model representation using DAGs. The goal of this set of experiments is to study computational savings that come from using a compact representation of individual (SDAG) or the full set (SDAG+) of cutting plane models in S. As the baseline for the learning and classification runtime comparison we use plain uSVM algorithm. To carry out training we use 100k of SRL dataset. The number of iterations for the algorithms is fixed at 300 and the rest of the parameters are kept at default values. For evaluating speedups obtained during classification phase we carry out learning on SRL subsets of increasing size and then test trained models on 10k of data.

[3] `http://l2r.cs.uiuc.edu/cogcomp/Data/QA/QC/`

[4] retrieved through the Yahoo! Webscope program.

Table 1. Runtime comparison between uSVM, SDAG and SDAG+. Training on 100k subset of SRL across multiple values of the sample size (left) and classification on 10k subset when learning on SRL subsets of varying size (right). Time indicated in seconds; values in parenthesis for SDAG and SDAG+ are speedups w.r.t uSVM; #SVs- number of support vectors.

Training			
sample	uSVM	SDAG	SDAG+
1000	2196	312 (7.0)	283 (7.8)
2000	8282	1127 (7.3)	752 (11.0)
3000	18189	2509 (7.2)	1275 (14.3)
4000	31012	4306 (7.2)	1802 (17.2)
5000	50060	6591 (7.6)	2497 (20.0)

Classification				
Data	#SVs	uSVM	SDAG	SDAG+
10k	1686	11	9 (1.1)	1 (24.0)
25k	3392	41	25 (1.6)	1 (33.2)
50k	5876	82	40 (2.1)	3 (28.3)
75k	7489	112	55 (2.0)	5 (20.5)
100k	8674	131	59 (2.2)	7 (19.5)

Runtime results for SDAG, SDAG+ and uSVM are reported in Table 1 (since SDAG and SDAG+ produce exact kernel evaluations, hence they train the same model as uSVM, accuracy is not of concern and is omitted). As one can see both SDAG and SDAG+ deliver significant speedups during the learning. SDAG+ is a clear winner here delivering speedups up to 20 when a large sample size is used. Regarding classification, SDAG+ is also the fastest, although as the number of support vectors of the learned model increases, the speedup factor decreases. This is due to the increased overhead of maintaining a single large DAG. As the number of elements in the DAG grows the inefficiencies from the implementation of the underlying data structure slow down the node lookup time. We plan to address this problem in the future.

Tuning up Precision and Recall. We first report experimental results on question classification corpus on six different categories in Table 2 (since the dataset is small, we only report the accuracy). For both uSVM and uSVM+j, we fixed the sample size to 100. For uSVM+j, we picked the value of j from $\{1, 2, 3, 4, 5, 10\}$ and use the best results obtained on the validation set. For SVM, we carried out tuning of j parameter on a validation set. It is important to note that such parameter has slightly different meaning for uSVM+j and SVM. For the former, it controls the bias to reject negative examples during sampling (Alg. 3) to compute MVC, while for the latter it defines the factor by which training errors on positive examples outweigh errors on negative examples.

Analyzing the results from Table 2 (top), we can see that uSVM algorithm that uses uniform sampling obtains high Precision, as it minimizes the training error dominated by examples from negative class. This results in lower values of the Recall. Its rather high F_1 for ABBR dataset shows that the model simply misclassifies the examples from the minority class saturating the Precision. On the other hand, uSVM+j is able to establish a much better balance between Precision and Recall resulting in high F_1 scores across the majority of categories. Also the performance of SVM with the optimal set of parameters suggests that our method has a better capacity to control the imbalance problem than SVM. This can be explained by the fact, as suggested in [22], that $z_i C$ imposes only an

Table 2. Handling class-imbalance problem on TREC 10 (top) YA (bottom). Ratio - proportion of negative examples w.r.t. positive; P/R - precision (P) and recall (R). The bottom row in YA is the performance using bag-of-words features on 75k subset.

Trec 10

Data	Ratio	uSVM		uSVM+j		SVM	
		F-1	P/R	F-1	P/R	F-1	P/R
ABBR	1:60	87.5	100.0/77.8	84.2	80.0/88.9	84.2	80.0/88.9
DESC	1:4	96.1	95.0/97.1	96.1	95.0/97.1	94.8	97.7/92.0
ENTY	1:3	72.3	91.8/59.6	79.1	79.6/78.7	80.4	82.2/78.7
HUM	1:3	88.1	98.1/80.0	90.3	94.9/86.2	87.5	88.9/86.2
LOC	1:3	81.4	96.6/70.4	87.0	87.5/86.4	82.6	86.5/79.0
NUM	1:5	86.0	98.9/76.1	91.2	96.1/86.7	89.9	98.9/82.3

Yahoo Answers

10k	1:1.5	37.4	33.5/42.2	39.1	29.6/57.7	37.9	24.2/87.7
50k	1:2.0	36.5	36.0/36.9	40.6	30.0/62.5	39.6	25.7/86.9
100k	1:2.4	33.4	36.2/31.1	40.2	30.2/59.9	40.3	26.6/83.5
150k	1:2.8	33.5	36.9/30.7	41.0	30.2/64.0	-	-
300k	1:3.4	23.8	40.1/16.9	41.4	30.7/63.8	-	-
BOW	1:2.0	34.2	33.2/35.3	38.1	27.5/61.7	36.3	22.5/93.5

upper bound on dual variables α_i, which results in poorer flexibility to control the class-imbalance with the j parameter of SVM.

The results on Yahoo! Answers are displayed in Table 2 (bottom). For uSVM and uSVM+j, we fix the sample size at 500. Due to the constant time scaling behavior of uSVM [23], the training time for both uSVM and uSVM+j was slightly less than 10 hours across all subsets reported here. While being faster on small subsets of 5k, 10k and 25k, SVM begins to scale poorly on the subsets larger than 50k. Indeed, as studied in [23,16], CPA with sampling begins to outperform SVM starting from datasets of moderate size (around 50k in our experiments). SVM did not finish the training within 5 days for 150k and 300k subsets, hence there are missing values. We set the value of j parameter for uSVM+j equal to the ratio of negative to positive examples. This natural setting of j parameter for uSVM+j is driven by the intuition to make the distribution of examples from different classes approximately balanced inside each sample, such that the classifier learns on a balanced data. As one can see, this gives much better trade-off between Precision and Recall compared to uSVM. Looking at the results of SVM, we conjecture that here j parameter, similar to the results in previous experiments, is not flexible enough to deliver the optimal P/R trade-off. Also note that training SVM on 100k subset requires almost 4 days, which makes uSVM+j a viable tool for advanced text classification on large datasets, where obtaining optimal balance between Precision and Recall is hindered by the class imbalance problem.

The bottom row of Table 2 reports the results using bag-of-words (BOW) feature representation on 75k subset. We note that SST kernel delivers an

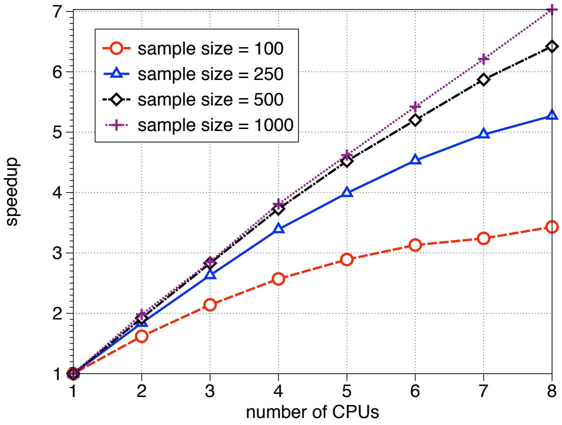


Fig. 2. Speedups due to parallelizing SDAG/SDAG+ on 50k Yahoo! Answers dataset

interesting 12% of relative improvement over BOW model on SVM. However, the main goal of this experiment was not to obtain the top classification performance on such noisy web data but rather to demonstrate that uSVM+j can efficiently deal with large imbalanced data.

Parallelization. To assess the effects of parallelization, we tested parallel versions of SDAG and SDAG+ on 50k subset of Yahoo! Answers dataset using up to 8 CPUs. The achieved speedups over the sequential algorithm are reported in Figure 2, where each curve corresponds to runtimes using different sample sizes: $\{100, 250, 500, 1000\}$. Increasing the sample size leads to the increase of the time spent to compute MVC, which makes the speedup achieved by parallelization for large sample sizes even more significant. Using the maximum number of 8 CPUs, we are able to achieve the speedup factor of about 7.0 (using sample size equal to 1000). Since classification can also be easily parallelized, we could experiment with larger sample sizes to obtain a more accurate model.

6 Related Work

To improve the scaling properties of SVM-light, a number of efficient algorithms using CPA-based algorithms have been proposed. For example, SVM^{perf} [10] exhibits linear computational complexity in the number of examples when linear kernels are used. While CPA-based approaches deliver state of the art performance w.r.t. accuracy and training time, they scale well only when linear kernels are used. The problem of efficient kernel learning for CPA has been studied in [11], where cutting plane models are compacted by extracting basis vectors. This, however, leads to a non-trivial optimization problem when arbitrary kernel functions are applied.

Regarding learning with structural kernels, compact representation of tree forests offered by DAGs was applied for speeding up training of the voted perceptron algorithm in [1]. Another interesting idea of hash kernels for structured data is proposed in [18], where hashing can generate explicit vector representation such that linear learning methods can be applied. However, it is likely that hashing all possible substructures generated by SST kernel, which is exponential in the tree length, will make the preprocessing step too expensive. Also, due to hash collisions, this method computes approximate kernel values and its implications on the accuracy need to be studied more extensively.

Concerning class-imbalance problem for SVM learning, the most widely adopted method is to introduce different cost factors in the objective function s.t. the training errors for positive and negative examples receive different penalties [21]. This approach is implemented as the j option in SVM-light [9] that has a super-linear scaling behavior, which prohibits its use on large datasets. Our approach to accomplish cost-sensitive classification shares the idea of reductions put forward in [24] together with the benefit of the conventional approach in SVMs [21] to incorporate importance weights directly into the optimization process.

7 Conclusions and Future Work

In this paper we have presented a set of techniques to make SVMs with structural kernels a more useful tool to apply in real-world tasks. First, we derive two learning algorithms SDAG and SDAG+ that compact cutting plane models using DAGs. This makes both learning and classification much faster when compared to the original CPA with sampling. Next, we present parallelized versions of both algorithms to deliver even faster runtimes. Finally, we propose an alternative sampling strategy to efficiently handle class-imbalanced data. The distinctive property of the proposed method is that it directly integrates the cost-proportionate sampling into the CPA optimization process, unlike the other sampling approaches based on the reductions idea of [24]. In other words, sampling is carried out iteratively, such that no information is discarded from training examples as in “one-shot” sampling methods.

Acknowledgements. This work has been partially supported by the EC project FP247758: Trustworthy Eternal Systems via Evolving Software, Data and Knowledge (EternalS).

References

1. Aiolli, F., Martino, G.D.S., Sperduti, A., Moschitti, A.: Efficient kernel-based learning for trees. In: CIDM, pp. 308–315 (2007)
2. Cancedda, N., Gaussier, E., Goutte, C., Renders, J.M.: Word sequence kernels. Journal of Machine Learning Research 3, 1059–1082 (2003)
3. Charniak, E.: A maximum-entropy-inspired parser. In: ANLP, pp. 132–139 (2000)
4. Collins, M., Duffy, N.: New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In: ACL, pp. 263–270 (2002)
5. Cumby, C., Roth, D.: Kernel Methods for Relational Learning. In: Proceedings of ICML 2003 (2003)

6. Daumé III, H., Marcu, D.: A tree-position kernel for document compression. In: Proceedings of the DUC, Boston, MA (May 6-7, 2004)
7. Fan, R., Chen, P., Lin, C.: Working set selection using the second order information for training svm. Journal of Machine Learning Research 6, 1889–1918 (2005)
8. Franc, V., Sonnenburg, S.: Optimized cutting plane algorithm for support vector machines. In: ICML, pp. 320–327 (2008)
9. Joachims, T.: Making large-scale SVM learning practical. In: Advances in Kernel Methods - Support Vector Learning, ch. 11, pp. 169–184. MIT Press, Cambridge (1999)
10. Joachims, T.: Training linear SVMs in linear time. In: KDD (2006)
11. Joachims, T., Yu, C.N.J.: Sparse kernel svms via cutting-plane training. Machine Learning 76(2-3), 179–193 (2009), eCML
12. Kate, R.J., Mooney, R.J.: Using string-kernels for learning semantic parsers. In: ACL (July 2006)
13. Kudo, T., Matsumoto, Y.: Fast methods for kernel-based text analysis. In: Proceedings of ACL 2003 (2003)
14. Moschitti, A.: Making tree kernels practical for natural language learning. In: EACL. The Association for Computer Linguistics (2006)
15. Palmer, M., Kingsbury, P., Gildea, D.: The proposition bank: An annotated corpus of semantic roles. Computational Linguistics 31(1), 71–106 (2005)
16. Severyn, A., Moschitti, A.: Large-scale support vector learning with structural kernels. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010. LNCS, vol. 6323, pp. 229–244. Springer, Heidelberg (2010)
17. Shen, L., Sarkar, A., Joshi, A.k.: Using LTAG Based Features in Parse Reranking. In: Proceedings of EMNLP 2006 (2003)
18. Shi, Q., Petterson, J., Dror, G., Langford, J., Smola, A.J., Vishwanathan, S.V.N.: Hash kernels for structured data. JMLR 10, 2615–2637 (2009)
19. Steinwart, I.: Sparseness of support vector machines. Journal of Machine Learning Research 4, 1071–1105 (2003)
20. 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)
21. Veropoulos, K., Campbell, C., Cristianini, N.: Controlling the sensitivity of support vector machines. In: Proceedings of the IJCAI, pp. 55–60 (1999)
22. Wu, G., Chang, E.: Class-boundary alignment for imbalanced dataset learning. In: ICML 2003 Workshop on Learning from Imbalanced Data Sets II, Washington, DC, pp. 49–56 (2003)
23. Yu, C.N.J., Joachims, T.: Training structural svms with kernels using sampled cuts. In: KDD, pp. 794–802 (2008)
24. Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In: Proceedings of ICDM (2003)

Generalized Agreement Statistics over Fixed Group of Experts

Mohak Shah

Accenture Technology Labs
161 N. Clark St., Chicago, IL, 60601, USA
mohak.shah@accenture.com

Abstract. Generalizations of chance corrected statistics to measure inter-expert agreement on class label assignments to the data instances have traditionally relied on the marginalization argument over a variable group of experts. Further, this argument has also resulted in agreement measures to evaluate the class predictions by an isolated classifier against the (multiple) labels assigned by the group of experts. We show that these measures are not necessarily suitable for application in the more typical fixed experts' group scenario. We also propose novel, more meaningful, less variable generalizations for quantifying both the inter-expert agreement over the fixed group and assessing a classifier's output against it in a multi-expert multi-class scenario by taking into account expert-specific biases and correlations.

Keywords: Agreement statistics, classifier evaluation, multiple experts.

1 Introduction

Performance evaluation of learning algorithms over data for which deterministic (true) labeling is unknown comes with unique issues. When the ground truth is known, the evaluation consists of calculating a loss function between the true labels and the ones predicted by the classifier. An indicator loss function is used in the case of classification algorithms resulting in an accuracy estimate. However, various relevant application scenarios exist where expert-labels are sought since the true labels cannot be determined due to one or more issues such as inadequate data acquisition, limited knowledge of the application domain and so on. Note that the *expert* can also be an automatic labeling process (e.g., a classifier). Further, labels from *multiple experts* are typically obtained to mitigate the variability in individual estimates[1]. Examples of such applications include tasks such as medical image segmentation or alignment of stock price movement prediction approaches with other market indicators (e.g., market analysts' predictions). With technological advances such as Amazon's Mechanical Turk[2]

[1] In this paper, we do not consider novice or extremely imperfect label generation processes, including experts.

[2] http://www.mturk.com

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 191–206, 2011.

(AMT), obtaining such labels for various human intelligence tasks is becoming increasingly easier.

It has been widely argued that chance agreements, resulting from experts' natural labeling propensities, should be taken into account while measuring inter-expert agreements. This is also the case when evaluating a classifier's performance, both against true (or even single expert generated) labels and against labels obtained from a group of experts. This argument resulted in various chance corrected agreement statistics such as Scott's π statistic (Scott 1955) and Cohen's kappa (Cohen 1960) measure (see (Kuncheva 2004, Japkowicz and Shah 2011) for discussion).

We consider the general form of this problem. Given a dataset S each of whose instances has been labeled by one of r experts, two quantities of interest need to be quantified. First, the extent of agreement *among* the r experts generating the labels, called the Inter-expert or Inter-rater agreement; and, Second, the extent to which the labeling output by a new classifier $\mathfrak{r}$ agrees with those of r experts as a group.

With regard to measuring the inter-expert agreement, however, the earlier attempts such as Cohen's kappa estimate resulted in statistics that applied only to *binary* classification scenarios over *two* sets of labels. One of the famous generalizations of Cohen's κ statistic was proposed by Fleiss (1971), denoted here by κ_F and has since been projected to be a standard in measuring inter-expert agreement (even hard coded in toolkits such as WEKA (Witten and Frank 2005)). Moreover, attempts motivated by argument along the lines of Fleiss (1971), although few, have also been made to quantify the agreement between a classifier (or an isolated expert) and a group of experts. One of the recent generalizations in this tradition has been that of Vanbelle and Albert (2009) that we will discuss later (we denote the unweighted variant $\hat{\kappa}$ in (Vanbelle and Albert 2009) here by κ_{va}).

In both these cases, the typical approach has been that of marginalizing over the experts comprising the group, under the *variable expert assumption* that each individual expert in the group comes from a (much larger) pool of experts and is interchangeable as long as the size of the group remains fixed. Marginalization over experts refers to obtaining probabilistic estimates of random assignment of a label to an example by a *random* expert. Since the experts need not be the same over instances in the variable expert assumption, this amounts to obtaining such estimates from the pool of all the labels by all the experts taken together. However, we contend that such estimation is not suitable over a *fixed* group of experts. In this case more information on correlated expert behavior is available and needs to be taken into account. By ignoring the expert specific biases and their correlations, the marginalized estimates invariably lead to pessimistic agreement measurements characterized by a higher variance in the *fixed expert group* scenario. In addition, the marginalization approach has more serious implications when there is a high heterogeneity in expert biases. Further, measures such as κ_{va}, motivated by similar arguments, also suffer from similar limitations when applied to the fixed expert scenario.

This paper proposes agreement statistics for measuring agreement *between* and *against* a group of r *fixed* experts. In particular, for inter-expert agreement, we propose a generalization of Cohen's kappa statistic to the case of multiclass classification (nominal scale) by a fixed group of experts. The proposed generalization has the property that it reduces to the classical version of Cohen's kappa in the case of binary classification by two experts. We then use this statistic to obtain a measure of agreement of a new classifier against the fixed group of experts. This argument results not only in tighter agreement statistics but also in more meaningful treatment of chance agreement as we will see below. A point to note here is that we do not assume existence of ground truth and as such neither attempt to learn the raters' behavior nor to obtain an estimate of the ground truth. Attempts along these lines have been made but differ in the inherent assumptions of the framework (see, for instance, (Raykar et al. 2010) which extends the STAPLE approach of (Warfield et al. 2004), or (Whitehill et al. 2009) that, based on different premise, propose estimating ground truth from multiple labels and also model the expertise of each labeler). This work also differs from the recent works in learning from crowds (e.g., (Snow et al. 2008)) settings in that we assume a setting in which the experts are assumed to be fixed and focus on obtaining evaluative estimates, as well as from works in developing probabilistic models (e.g., (Yan et al. 2010)) on annotater expertise to provide an estimate of true label in that we do not assume a determinable ground truth.

The rest of the paper is organized as follows: Section 2 proposes a new measure for inter-expert agreement estimation by treating chance agreement in a more coherent manner w.r.t. the observed agreement. Based on this, Section 3 then introduces a novel measure to estimate agreement between a classifier and a fixed group of experts. Both this sections also contrast the proposed measures with their respective marginalization-based counterparts. Section 4 provides an insight into both asymptotic and empirical behavior of the proposed statistics along with the crucial differences from related metrics. These insights are empirically supported by some results on synthetic and real data in Section 5. Section 6 discusses some related approaches along with their limitations with regard to fixed expert group scenario. Finally, we conclude in Section 7.

2 Measuring Inter-expert Agreement

Let $S = \{\mathbf{i}_1, \ldots, \mathbf{i}_n\}$ denote a dataset with n instances. Each instance $\mathbf{i} \in S$ is assigned one of the k class labels from $\left\{l^1, \ldots, l^k\right\}$ by a group $\mathcal{R}$ of r experts. By c_{ij} we denote the number of experts assigning instance $\mathbf{i}_i$ to class l^j. Also, c_{pj} denotes the number of instances assigned to class l^j by expert p. Note that the measures such as κ_F (as well as κ_{va}) that marginalize over experts assume $\mathcal{R} \subset \mathfrak{R}$ where $\mathfrak{R}$ denotes a pool of experts from which $\mathcal{R}$ is drawn for different instances. However, under our setting $\mathcal{R}$ is considered to be fixed as is the case in many typical applications of the kind mentioned above.

Given any agreement statistic $\mathbb{A}$, a chance corrected agreement estimate can be defined as $\kappa = \frac{\mathbf{E}_S(\mathbb{A}) - \mathbf{E}(\mathbb{A})}{\max_S(\mathbb{A}) - \mathbf{E}(\mathbb{A})}$ where $\mathbf{E}_S(\mathbb{A})$ denotes the average empirical agreement between the experts on dataset S, $\mathbf{E}(\mathbb{A})$ denotes the true expectation of $\mathbb{A}$ and $\max_S(\mathbb{A})$ denotes the maximum achievable agreement between the experts on dataset S. Various characterizations of the agreement statistic $\mathbb{A}$ lead to different agreement estimates. For instance, Cohen's κ assumes $\mathbb{A}$ to be proportion of instances on which two raters agree in a binary classification scenario. Consequently, the true expectation measures the probability that these raters will agree just by chance (based on their individual labeling probabilities/biases) on a random example. In this sense, different agreement estimates attempt to capture different characteristics of the scenario to obtain assessments of rater agreements obtained above and beyond their coincidental concordance, also referred to as chance agreement (the expectation).

For notational simplicity, let us denote $\mathbb{A}_o = \mathbf{E}_S(\mathbb{A})$, $\mathbb{A}_e = \mathbf{E}(\mathbb{A})$ and $\mathbb{A}_{max} = \max_S(\mathbb{A})$. Hence,

$$\kappa = \frac{\mathbb{A}_o - \mathbb{A}_e}{\mathbb{A}_{max} - \mathbb{A}_e} \tag{1}$$

We adopt a pairwise agreement statistic for $\mathbb{A}$ to model the expert agreements. By taking into account the agreement between all the individual pairs of experts, we can quantify the overall observed agreement among the r experts as:

$$\mathbb{A}_o = \frac{1}{n}\sum_{i=1}^{n} A_o(\mathbf{i}_i) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} A_o(\mathbf{i}_i, l^j) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\frac{1}{r(r-1)}\sum_{p\in\mathcal{R}}\sum_{p'\in\mathcal{R},p'\neq p} e_{ij}^{p}\cdot e_{ij}^{p'}$$

where $A_o(\mathbf{i}_i, l^j)$ is nothing but the proportion of pairwise agreement between experts over an instance $\mathbf{i}_i$ assigned to class l^j out of a total of $r(r-1)$ possible expert pairs; $e_{ij}^{p} = 1$ if the expert p assigns instance $\mathbf{i}_i$ to class l^j and zero otherwise. Notice that this includes the duplicate pairs of experts as well. However, we adhere to this more general form since potentially the pairwise costs may be asymmetric. Weights to take into account these asymmetric costs can then be easily integrated in this form.

The above computation yields:

$$\mathbb{A}_o = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\left[\frac{1}{r(r-1)}c_{ij}(c_{ij}-1)\right] \tag{2}$$

This agreement criterion has been widely utilized including the case of κ_F. The measures assuming a variable expert case obtain the chance agreement by relying on marginalization over the experts. For instance, Fleiss (1971) used $\mathbb{A}_e = \sum_{j=1}^{k} c_{.j}^2$ where $c_{.j} = \frac{1}{nr}\sum_{i=1}^{n} c_{ij}$. Replacing this in Equation 1 and setting $\mathbb{A}_{max} = 1$ gives the κ_F coefficient. However, this results in an excessively optimistic estimate over chance agreement which in turns gives a very conservative agreement statistic. For unique pairs of experts, the expectation of the pairwise $\mathbb{A}$ statistic has been discussed by (Hubert 1977). We use a similar argument for the more general case of all possible pairs used in $\mathbb{A}_o$ of Equation 2 and obtain, for the fixed experts case:

$$\mathbb{A}_e = \sum_{j=1}^{k} A_e(l^j) = \sum_{j=1}^{k} \frac{1}{r(r-1)} \sum_{p \in \mathcal{R}} \sum_{p' \in \mathcal{R}, p' \neq p} \left[v_j^p v_j^{p'} \right] \tag{3}$$

where $A_e(l^j)$ quantifies the pairwise chance agreement between the experts on any given class label l^j such that v_j^p denotes the probability with which the expert p assigns a random instance to class l^j. The empirical estimate of $\mathbb{A}_e$ can be obtained from the data as:

$$\mathbb{A}_e = \sum_{j=1}^{k} \frac{1}{r(r-1)} \sum_{p \in \mathcal{R}} \sum_{p' \in \mathcal{R}, p' \neq p} \left[\frac{c_{pj}}{n} \frac{c_{p'j}}{n} \right] \tag{4}$$

Using $\mathbb{A}_o$ and $\mathbb{A}_e$ defined in Equations 2 and 4 respectively, along with $\mathbb{A}_{max} = 1$ in Equation 1, we get the desired inter-expert agreement statistic as:

$$\kappa_S = \frac{\sum_{i=1}^{n} \sum_{j=1}^{k} c_{ij} \cdot (c_{ij} - 1) - \frac{1}{n} \sum_{j=1}^{k} \sum_{p \in \mathcal{R}} \sum_{p' \in \mathcal{R}, p' \neq p} [c_{pj} c_{p'j}]}{nr(r-1) \Big[1 - \frac{1}{n^2 r(r-1)} \sum_{j=1}^{k} \sum_{p \in \mathcal{R}} \sum_{p' \in \mathcal{R}, p' \neq p} [c_{pj} c_{p'j}] \Big]} \tag{5}$$

Note that this statistic reduces to the classical version of Cohen's Kappa for the case of $k = r = 2$.

3 Measuring Agreement against a Group of Experts

Let $\mathfrak{r}$ denote a new classifier (or expert) with $\mathfrak{r}_{ij}$ being unity if $\mathfrak{r}$ assigns a label l^j to instance $\mathbf{i}_i$ and zero otherwise. Since we assume $\mathfrak{r}$ to be a discrete classifier, $\mathfrak{r}_{ij}$ can be interpreted in a probabilistic sense.

In this discrete classification scenario, one way to measure the agreement of $\mathfrak{r}$ against $\mathcal{R}$ would be to measure the agreement of label assigned by $\mathfrak{r}$ against the proportion of experts from $\mathcal{R}$ over certain class of interest l^j while controlling for other classes (that is, mapping it to a binary problem by assigning 1 to l^j and 0 to all other classes), in a fashion similar to the IntraClass kappa Coefficient (ICC) (Kraemer 1979). Similarly, an empirical estimate over the expectation of this statistic (the chance agreement) could be obtained by marginalizing over this proportion (under variable assumption of $\mathcal{R}$ sampled from $\mathfrak{R}$ for each instance) in conjunction with the label assigned by $\mathfrak{r}$. Using these and then adjusting for the maximum achievable agreement can give us the extent to which $\mathfrak{r}$ agrees with $\mathcal{R}$. This one-against-all approach can then be extended to multi-class scenario by iterating over classes with a different l^j being set to 1 in each iteration. This is the approach adopted by Vanbelle and Albert (2009), referred to here as κ_{va}. However, the problems with this approach in the fixed experts' group scenario, are obvious. First is, of course, related to its formulation which ignores interaction biases over classes other than the class of interest since these classes are lumped together while mapping the problem to the binary case (although the parameter estimation takes the general form and is directly computable). Moreover, due to the implicit assumption over variable $\mathcal{R} \in \mathfrak{R}$, it marginalizes over the expert biases. We propose an alternate measure for the fixed $\mathcal{R}$ case based on the consideration of the κ_S measure derived above.

Extending our notion of pairwise agreement between experts in $\mathcal{R}$ on each example, the overall observed agreement between $\mathfrak{r}$ and $\mathcal{R}$ can be obtained as:

$$\mathbb{A}_o = \mathbf{E}_{\mathbf{i}\sim S}[\mathbb{A}_o(\mathbf{i}_i)] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{A}_o(\mathbf{i}_i) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\mathfrak{r}_{ij}A_o(\mathbf{i}_i, l^j) \tag{6}$$

where $\mathbb{A}_o(\mathbf{i}_i)$ is the agreement between $\mathfrak{r}$ and the group of raters over all the classes, with $A_o(\mathbf{i}_i, l^j)$ as defined in Equation 2.

Next, $\mathbb{A}_e$ for this case denotes the overall chance agreement between $\mathfrak{r}$ and the group of experts $\mathcal{R}$. Its empirical estimate can be obtained analogous to our previous discussion as:

$$\mathbb{A}_e = \sum_{j=1}^{k}\mathfrak{r}_j \cdot A_e(l^j) = \sum_{j=1}^{k}\left[\mathfrak{r}_j \cdot \frac{1}{r(r-1)}\sum_{p\in\mathcal{R}}\sum_{p'\in\mathcal{R},p'\neq p}\left[\frac{c_{pj}}{n}\frac{c_{p'j}}{n}\right]\right] \tag{7}$$

where $\mathfrak{r}_j = \mathbf{E}_{\mathbf{i}\sim S}\mathfrak{r}_{ij} = \frac{1}{n}\sum_{i=1}^{n}\mathfrak{r}_{ij}$ is the probability of the rater $\mathfrak{r}$ classifying a random example to class l^j.

The next important quantity to evaluate is the maximum achievable agreement $\mathbb{A}_{max}$, *between* $\mathfrak{r}$ and the group of experts $\mathcal{R}$. Note that the earlier measures assumed this to be unity (see, e.g., (Schouten 1982)). This turns out to be a significant limitation since $\mathbb{A}_{max} = 1$ if and only if *all* the raters in $\mathcal{R}$ agree on *all* the instance labels and further when $\mathfrak{r}$ agrees with these labels on all the instances. That is, this assumes κ_S to be unity as a pre-condition for the measure to achieve perfect score. However, this does not reflect the goal of assessing the extent of agreement of the classifier labeling with that of the group $\mathcal{R}$.

This is a crucial point. What we are interested in is the maximum agreement that the classifier $\mathfrak{r}$ can achieve against the *combined* labelings of $\mathcal{R}$ *independent of* the extent of agreement achieved among experts in $\mathcal{R}$ (of course still requiring at least some degree of inter-expert agreement for group qualification). Hence, the maximum agreement that $\mathfrak{r}$ can achieve would be when it assigns a labeling such that each assigned label corresponds to the label on which there is a maximum agreement *actually obtained* among the experts in $\mathcal{R}$.

With this consideration, we define the maximum possible agreement in our case as:

$$\mathbb{A}_{max} = \frac{1}{n}\sum_{i=1}^{n}\max_j A_o(\mathbf{i}_i, l^j) = \frac{1}{n}\sum_{i=1}^{n}\max_j\left(\frac{1}{r(r-1)}(c_{ij}(c_{ij}-1))\right) \tag{8}$$

Hence, replacing $\mathbb{A}_o$, $\mathbb{A}_e$ and $\mathbb{A}_{max}$ from Equations 6, 7 and 8 in Equation 1, the new measure, denoted as $\mathcal{S}$, of agreement between a classifier and a fixed group of experts becomes:

$$\mathcal{S} = \frac{\frac{1}{n}\sum_{i=1}^{n}\left[\sum_{j=1}^{k}\mathfrak{r}_{ij}A_o(\mathbf{i}_i, l^j)\right] - \sum_{j=1}^{k}\mathfrak{r}_j \cdot A_e(l^j)}{\frac{1}{n}\sum_{i=1}^{n}\max_j A_o(\mathbf{i}_i, l^j) - \sum_{j=1}^{k}\mathfrak{r}_j \cdot A_e(l^j)}$$

Notice that, unlike the measure of Vanbelle and Albert (2009), $\mathcal{S}$ enables incorporating the expert-specific bias in calculating both $\mathbb{A}_e$ and $\mathbb{A}_{max}$ by considering $\mathcal{R}$ to be fixed. Also, $\mathcal{S}$ can be evaluated directly over all classes unlike the former.

4 Analysis

We present another main result of this work in the form of a theorem upper bounding the variance of the proposed κ_S statistic and showing how this is a more stable measure than κ_F in the fixed-experts case. The arguments for the variance analysis for various agreement measures follow from the large sample estimation of moments in the statistics literature (see, for instance, (Rao 2001)). While we relegate the detailed analysis of empirical variances and associated statistical significance tests for these statistics to the longer version of the paper, we nevertheless deem it important to discuss the following theoretical result.

Theorem 1. *Let κ_F and κ_S denote, respectively, the agreement statistics of Fleiss (1971) and that proposed in Equation 5 computed on a population (dataset) with large sample-size n where each of the sample has been assigned one of k labels by a fixed group of r experts. If $\sigma^2(\kappa)$ denotes the variance of κ then we have that:*

$$\sigma^2(\kappa_S) \leq \sigma^2(\kappa_F)$$

with equality satisfied when the experts emulate the pool.

Proof. The hypothesis of no agreement suggests labeling according to $\mathbf{E}(\mathbb{A})$ (or $\mathbb{A}_e$). Let us analyze this chance agreement $\mathbb{A}_e$ with regard to κ_S as defined in Equation 4. Under our formulation we can model the bias of each expert assigning example $\mathbf{i} \in S$ to a class $l^j, j \in \{1, \ldots, k\}$ as a multinomial b. That is, the multinomial $b_p(l^j)$ models the probability with which the expert p assigns a label l^j to a random example $\mathbf{i}$ chosen from S. The overall bias of expert p can then be modeled by a vector $\mathbf{b}_p = (b_p(l^1), \ldots, b_p(l^k))$. Hence, the chance agreement $\mathbb{A}_e$ essentially models these probabilities of the pairs of experts for each class which for the purposes of our analysis can be considered a constant. Therefore, the variance of κ_S depends basically on the variance of the observed agreement $\mathbb{A}_o$ of Equation 2. Then for large samples, for any agreement statistic $\mathbb{A}$, the variance of the metric $\kappa = \frac{\mathbf{E}_S(\mathbb{A}) - \mathbf{E}(\mathbb{A})}{\max(\mathbb{A}) - \mathbf{E}(\mathbb{A})}$ can be obtained as:

$$\sigma^2(\kappa) = \frac{\sigma^2(\mathbb{A})}{[\max(\mathbb{A}) - \mathbf{E}(\mathbb{A})]^2}$$

For the case of κ_S statistic, disregarding the constants, the expectation of the agreement statistic (denoted with superscript κ_S), is:

$$\mathbf{E}(\mathbb{A}^{\kappa_S}) = \sum_{j=1}^{k} \frac{1}{r(r-1)} \sum_{p \in \mathcal{R}} \sum_{p' \in \mathcal{R}, p' \neq p} \left[v_j^p v_j^{p'} \right] \tag{9}$$

Consequently, the variance becomes:

$$\sigma^2(\mathbb{A}^{\kappa_S}) = \sum_{p \in \mathcal{R}} \sum_{p' \in \mathcal{R}, p' \neq p} \left[\sum_{j=1}^{k} (v_j^p v_j^{p'} (1 - v_j^p - v_j^{p'})) + (\sum_{j=1}^{k} v_j^p v_j^{p'})^2 \right] \tag{10}$$

Similarly, for the case of κ_F, without differentiating between the experts (under the variable expert assumption) and disregarding constants, the expectation of the agreement term becomes,

$$\mathbf{E}(\mathbb{A}^{\kappa_F}) = \sum_{j=1}^{k} c_{.j}^2 \tag{11}$$

Now, the variance $\sigma^2(\mathbb{A}^{\kappa_F})$, using Equation 2 for $\mathbb{A}_o$, can be approximated as:

$$\sigma^2(\sum_j c_{ij}^2) = 2r(r-1)[\sum_j (c_j^2) - (2n-3)(\sum_j (c_j^2))^2 + 2(n-2) \sum_j (c_j^2)] \tag{12}$$

Note, however, the crucial difference between the no agreement hypotheses assumed by κ_S and κ_F. In the case of former, we assume that experts label the instances according to their respective biases while in the case of latter, we assume that the labeling occurs in agreement with the marginals of the pool of experts.

Hence, it can be seen that the expectation $\mathbf{E}(\mathbb{A}^{\kappa_S})$ of Equation 9 is upper bounded by $\mathbf{E}(\mathbb{A}^{\kappa_F})$ of Equation 11. Similarly, Equation 12 upper bounds Equation 10. Now, since both κ_S and κ_F consider all possible expert pairs, the constants would be identical, i.e. $n^2 r^2 (r-1)^2$ in the denominator for the variance calculation of both measures. This concludes the proof. □

Using similar arguments, the variance of $\mathcal{S}$ can be seen to be upper bounded by the variance of κ_{va}. The sampling variances for $\mathcal{S}$ and κ_{va} can be computed using the Jackknife or leave-one-out method (Efron and Tibshirani 1993, Japkowicz and Shah 2011). For $\mathcal{S}$, let $\mathcal{S} \backslash i$ denote the agreement on the label assignments of all the instances in S except $\mathbf{i}_i$. Calculating $\mathcal{S} \backslash i$ repeatedly n times leaving a different instance each time and subsequently averaging it can then yield the pseudovalues.

4.1 Properties and Behavior

The marginalization argument for estimating $\mathbb{A}_e$, such as that in κ_F, can result in excessively pessimistic agreement estimates. That is, while such measures estimate the observed *agreement*[3] they do not measure the chance probabilities over agreements. As a result the inter-expert correlations, partly as a result of the variable experts assumption, are ignored. This not only results in a loose estimate of $\mathbb{A}_e$ but can also yield less meaningful (even unwarranted negative)

[3] The pairwise consideration highlight that it would take at least 2 experts to agree on any given instance for the observed agreement to be non zero.

values of agreement measure even when the empirical evidence is to the contrary, for smaller values of $\mathbb{A}_o$. We will illustrate this in the next Section. In the fixed expert case, κ_S offers better consistency in the estimation of $\mathbb{A}_o$ and $\mathbb{A}_e$.

Similarly, while κ_{va} depends on the proportion of experts with maximum labels when computing $\mathbb{A}_{max}$, $\mathcal{S}$ depends on the labels on which the pairwise agreement over $\mathcal{R}$ is maximum. Hence, even when all the experts disagree over labels for all the instances, $\mathbb{A}_{max}$ is not zero for κ_{va} while it is zero in the case of $\mathcal{S}$. The latter is indeed desirable in the fixed expert group case since in the event of no agreement among the experts themselves (extreme variability), the agreement of the classifier with any individual expert, being unrepresentative of the *group* agreement, is rendered meaningless (even more so when $k > r$). Similar differences exist in the computation of other quantities. The marginalization argument when applied to the case of calculating $\mathbb{A}_e$ can also result in an overly conservative, and sometimes less meaningful, estimates of κ_{va} in contradiction with the the empirical evidence (as we will see in the next Section). There is an important point to be made here. While, analytically, it can be seen that κ_F is more conservative than κ_S, such a relationship need not exist between κ_{va} and $\mathcal{S}$ since their estimates would depend not only on the expert labels but also on their subsequent agreement with the new classifier. The contribution of individual expert (even when it disagrees with all the others) is not zero for both κ_F and κ_{va}.

5 Empirical Results and Discussion

We compare the behavior of the most commonly employed κ_F metric for inter-expert agreement measurement against the proposed measure κ_S. With regard to estimating the agreement of a classifier with group of fixed experts, we compare the generalization proposed by Vanbelle and Albert (2009) denoted as κ_{va} with the proposed $\mathcal{S}$ metric.

For both sets of comparisons, we use 4 different sized multi-class datasets from UCI repository (Asuncion and Newman 2007) over WEKA implementations of 6 different classifiers in addition to their true labels. Further, we also illustrate the limitations of κ_F in the fixed experts case with the help of synthetic data. The main aims of the simulations presented here are two-fold: i) illustrating the differences between the compared measures; and ii) highlighting the discrepancies in the estimation of variable expert assumption based measures when applied to the fixed experts cases making them unsuitable for the purpose. The datasets used include CMC (1473 instances, 3 classes), Car (1728 instances, 4 classes), Iris (150 instances, 3 classes) and Glass (214 instances, 7 classes) while the learning algorithms used are Support Vector Machine (with linear kernel), Naive Bayes, C4.5 Decision Trees, 3-Nearest Neighbor, Ripper and a Conjunction Rule learning algorithm. The reported results are over 10-fold Cross Validation with default parameter values over learning algorithms[4].

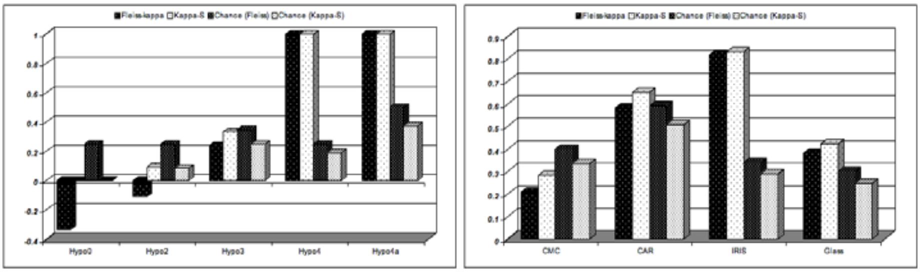
[4] Note that model selection is not our main concern here since we aim to show the difference in the agreement estimates.

Finally, we illustrate these differences in a real world example of Syphilis Serogen data (Williams 1976). Novel venues such as AMT can also yield relevant data for such simulations. While data from learning from crowds scenario are sometime publicly available (Snow et al. 2008), we are not aware of any relevant AMT datasets available yet for the fixed experts case.

5.1 Evaluating Inter-expert Agreement

Let us first consider a simple synthetic dataset of 200 instances labeled by 4 experts (E1, E2, E3 and E4) into one of the 4 classes (L1, L2, L3 and L4) and consider 5 different illustrative scenarios. The first label configuration "Hypo0" denotes the case when on each instance all four experts disagree. We do this by simply making E1 assign L1, E2 assign L2 and so on to all instances. Next, we flip the first 100 labels of E1 from L1 to L2 and the last 100 labels of E2 from L2 to L1 so that these two experts agree on all the labels while still disagreeing with E3 and E4 who themselves are in disagreement. This case is denoted "Hypo2". Next, from "Hypo2", we let E3 assign L1 to the first 100 instances and L2 to last 100 instances so that E3 agrees with both E1 and E2 yielding dataset "Hypo3". We then obtain a dataset "Hypo4" where all experts assign L1, L2, L3 and L4 to the respective subsets of 50 instances and are in complete agreement. Finally, We obtain a dataset "Hypo4a" that too simulates all experts in agreement but this time all the experts assign L1 to the first 100 instances and L2 to last 100 instances. (Note that the suffix in the name of each synthetic variation denote the number of experts in complete agreement). The results are presented in Figure 1(a).

In the case of "Hypo0", $\mathbb{A}_e > 0$ for κ_F so that $\kappa_F < 0$. Note here that there is a strict heterogeneity among experts' biases in this case (e.g. E1 never assigns any label other than L1, and so on) and the estimation of $\mathbb{A}_e$ using the marginalization argument is not meaningful since it does not reflect the probabilities of random label assignments. Unlike κ_F, $\mathbb{A}_e = 0$ for κ_S. Similarly, the marginalization argument over $\mathbb{A}_e$ results in negative value for κ_F in the



(a) Synthetic Cases

(b) UCI Data

Fig. 1. Comparison of the proposed κ_S (kappa-S in figures) statistic against the Fleiss' Kappa coefficient κ_F. Also shown are the corresponding estimates of chance agreements in both cases. An absence of a bar indicates a zero value.

event of partial agreement in the case of "Hypo2" data where E1 and E2 agree on all the instances. Again, this is not desirable since, here, both the $\mathbb{A}_o$ and $\mathbb{A}_e$ are solely based on E1 and E2 for classes L1 and L2. Keeping in view the label assignment in these cases, κ_S gives a more realistic estimate. Note that $\kappa_F \rightarrow \kappa_S$ as $\mathbb{A}_o \rightarrow 1$. This can be seen in the case of "Hypo4" and "Hypo4a" datasets. However, even when $\mathbb{A}_o$ for both the measures is 1, the chance agreement is not treated in the similar manner in these two cases.

Next, we compare the inter-expert agreement between the set of 7 label sets obtained on the 4 UCI datasets, one from each of the 6 classifiers, and the true labels of the instances, using both the κ_F and κ_S measures in Figure 1(b). As a result of optimistically estimating the chance agreement by marginalization over experts, κ_F is consistently more conservative than κ_S. However, the measures seem to converge with increasing levels of agreement with $\kappa_F \rightarrow \kappa_S$ as the agreement approaches unity. An example can be seen in the case of Iris datasets where classifiers typically obtained a very high accuracy rate and are in high agreement. However, the gap between the two measures is higher for moderate to low agreement values.

5.2 Evaluating Agreement against a Group of Fixed Experts

We consider the UCI datasets to compare $\mathcal{S}$ against κ_{va}. For each case, the experts' group is simulated by taking into account the true labels along with the two classifiers achieving highest 10-fold accuracy on each dataset. The (unweighted) κ_{va} and $\mathcal{S}$ are then estimated for each of the remaining classifiers against the group. The results are presented in Figure 2. As can be seen, κ_{va} consistently results in a conservative agreement estimate as compared to $\mathcal{S}$ in

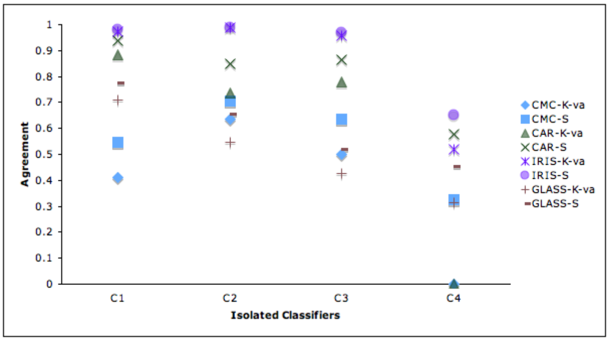


Fig. 2. Comparison of the proposed $\mathcal{S}$ measure against the κ_{va} statistic for measuring the agreement of isolated classifier (on horizontal axis) against expert labels composed of the actual labels and outputs of SVM and Decision Trees for the case of Car, CMC and Iris datasets; and of actual labels along with outputs of Ripper and 3-NN in the case of Glass dataset. C1 and C4 represent 3-NN and Conjunction Rule respectively in all datasets. C2 and C3 represents, respectively, SVM and NB in the case of Glass data while NB and Ripper in the case of the other three datasets.

these cases (at least partly due to marginalization). Note, in particular, the case of Car and CMC datasets over the comparison of conjunction rule learner against expert labels. While in both cases Conjunction rule learner obtains a trivial classifier assigning class 1 to all the instances, it should be noted that this class is highly overrepresented in these datasets (about 70% in CAR and 42% in CMC). While κ_{va} gives a less meaningful null estimate in both these cases, this scaling is better captured by the $\mathcal{S}$ measure as can be seen in the last column of Figure 2. Also, the two measures seem to converge as $\mathbb{A}_o$ approaches unity (see e.g., the Iris dataset over C1, C2 and C3 in Figure 2).

5.3 Illustration on Real Data

We illustrate the proposed indices of agreement on the Syphilis Serogen data of Williams (1976) who presented result obtained by three reference laboratories (denoted by Ref-1, Ref-2 and Ref-3) and an additional participant laboratory (denoted by T) on 28 specimens (data is shown in Table 1). Each specimen

Table 1. Syphilis Serogen data of (Williams 1976) used in Section 5

#	**T**	**Ref-1**	**Ref-2**	**Ref-3**
1	RE	RE	RE	RE
2	RE	RE	RE	RE
3	BL	NR	NR	NR
4	BL	NR	NR	NR
5	BL	NR	NR	NR
6	RE	RE	RE	RE
7	BL	NR	NR	NR
8	RE	RE	RE	RE
9	NR	NR	NR	NR
10	NR	NR	NR	NR
11	RE	RE	RE	RE
12	RE	RE	BL	BL
13	RE	RE	RE	RE
14	RE	RE	BL	BL
15	RE	RE	RE	RE
16	RE	RE	NR	BL
17	RE	RE	NR	BL
18	RE	RE	RE	RE
19	RE	RE	RE	RE
20	BL	BL	NR	NR
21	RE	RE	RE	RE
22	BL	NR	NR	NR
23	BL	BL	NR	NR
24	BL	BL	NR	NR
25	RE	RE	RE	RE
26	NR	NR	NR	NR
27	RE	RE	RE	RE
28	NR	NR	NR	NR

was classified into one of the three classes viz. Non Reactive (NR), Borderline (BL) and Reactive (RE). The additional participant laboratory also classified the 28 specimen into one of the three classes. The same dataset was also used by Vanbelle and Albert (2009, Table 3). For both κ_F and κ_S, we get $\mathbb{A}_o = 0.81$ between the three reference laboratories. The difference, analogous to the synthetic cases, appears in terms of optimistic estimate of $\mathbb{A}_e = 0.412$ in the case of κ_F. In the case of κ_S, we obtain $\mathbb{A}_e = 0.272$. Hence, we obtain $\kappa_F = 0.676$ and $\kappa_S = 0.738$. We can see how κ_F results in a pessimistic agreement estimate due to overestimating chance agreement. Also, note the difference in the results obtained for κ_S as compared to the agreement statistic such as ICC which was found to be 0.68 as reported by Vanbelle and Albert (2009).

Similarly, when comparing the three reference laboratories to laboratory T, we obtain, for κ_{va}: $\mathbb{A}_o = 0.655$, $\mathbb{A}_e = 0.362$ and $\mathbb{A}_{max} = 0.893$. On the other hand, for $\mathcal{S}$, we get: $\mathbb{A}_o = 0.571$, $\mathbb{A}_e = 0.105$ and $\mathbb{A}_{max} = 0.81$. This gives $\kappa_{va} = 0.551$ and $\mathcal{S} = 0.662$. These results too demonstrate the manner in which the two measures differ in the estimation of various quantities.

6 Related Work

In addition to Fleiss' coefficient, various other general inter-expert agreement measures such as the well known ICC (Kraemer 1979) or context-specific measure of Schouten (1982), have also appeared (e.g., see (Kuncheva 2004) for a discussion on some such measures in the context of classifier fusion). However, these measures too typically marginalize over the experts.

In this respect the proposed κ_S statistic is more in line with the arguments of Berry and Mielke Jr (1988) who propose a generalization over interval measurements and multiple experts by way of measuring the extent of disagreements between the experts in the l_2-norm setting. However, the disagreement measured by the Δ function there need not reflect the corresponding agreement under the l_2-norm. Furthermore, it requires rescaling the label assignments.

With regard to measuring the agreement against the group of experts, another commonly applied approach is the consensus approach where, for each instance, the label assigned by a majority (defined by a consensus threshold) of the experts is considered as the true label (see, for instance, (Soeken and Prescott 1986, Smith et al. 2003)). This simplifies the subsequent evaluation against a classifier by mapping the problem to a deterministic label matching problem amenable to more conventional techniques such as Cohen's kappa. However, such consensus labeling makes the output dependent on the consensus threshold and has serious limitations. In addition, issues such as not accounting for experts' dispersion as well as difficulty in dealing with instances with no consensus makes this approach highly contentious (Eckstein et al. 1998, Salerno et al. 2003, Miller et al. 2004).

Approaches proposed to bypass such consensus requirement such as those of Schouten (1982), Williams (1976) and Light (1971) either do not address the problem of interest directly or pose issues such as introduction of bias or ignoring interdependence of experts (Vanbelle and Albert 2009). Note, in particular, that the approach of Schouten (1982), even though applied in fixed expert settings, disregards the interdependence of experts when measuring agreement of one expert against others in the *same* group.

Another important caveat in above approaches lies in the assumption over the maximum achievable agreement between the classifier and the group of experts being unity. This caveat has profound implications since it makes the assessment of classifier performance *dependent on the inter-expert agreement*. Such measures, hence, can achieve a perfect score for the classifier *only when* the inter-expert agreement is unity which essentially obviates the need for (and utility of) multiple experts altogether. Vanbelle and Albert (2009) also noted these limitations and proposed an alternative general measure (κ_{va} discussed earlier). As mentioned above, κ_{va} too followed the marginalization argument in a binary classification case. It was then extended to the multi-class case in an indirect manner using an iterative one-against-all strategy[5]. We compared κ_{va} against $\mathcal{S}$ on various criteria above.

7 Conclusion and Future Work

In this paper, we noted the main limitations of measures based on marginalization over experts rendering them unsuitable for application in the typical fixed experts' group scenario. Among the crucial issues lie the excessively conservative agreement estimate obtained by the inter-expert agreement measures such as κ_F. Moreover, these measures, as seen in both theoretical arguments and empirical results, can yield less meaningful values when the heterogeneity in the expert biases is high. We also proposed two novel statistics, respectively, to measure inter-expert agreement (κ_S) *between*, and agreement of a classifier *against*, a fixed group of experts ($\mathcal{S}$) in the general case of multiple classes and multiple experts. The main advantage of the proposed measures can be seen in terms of their accounting for expert specific biases and correlations yielding tighter agreement assessments. The proposed measure $\mathcal{S}$ also scales the maximum achievable agreement in accordance thereby allowing more meaningful characterization of classifier's performance that is *independent* of the agreement achieved *within* the expert group. Finally, in contrast to the marginalization based measures, κ_S reduces to the classical Cohen's κ in the binary classification case over two label sets. The future work includes investigating the behavior and dependence of proposed statistics, as well as extending them, to testing scenarios such as asymmetric loss, bias, prevalance and class imbalance. Another area worth investigating is the sample size requirement for the data over classes since the expert specific biases are obtained from the data empirically. A sparse class can in principle affect such estimates adversely (of course, even in this case, the assessed biases are best that can be obtained in accordance with both the maximum likelihood as well as information-theoretic arguments). Finally, the proposed measures can also be generalized for probabilistic classifiers.

[5] This effectively generalizes Fleiss' kappa, or alternatively Scott's π statistic and not Cohen's kappa.

References

Asuncion, A., Newman, D.J.: UCI machine learning repository. University of California, School of Information and Computer Science, Irvine, CA (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html

Berry, K.J., Mielke Jr, P.W.: A generalization of cohen's kappa agreement measure to interval measurement and multiple raters. Educational and Psychological Measurements 48, 921–933 (1988)

Cohen, J.: A coefficient of agreement for nominal scales. Educational and Psychological Measurements 20, 37–46 (1960)

Eckstein, M.P., Wickens, T.D., Aharonov, G., Ruan, G., Morioka, C.A., Whiting, J.S.: Quantifying the limitation of the use of consensus expert commitees in roc studies. In: Proceedings SPIE: Medical Imaging 1998, vol. 3340, pp. 128–134 (1998)

Efron, B., Tibshirani, R.J.: An introduction to the bootstrap. Chapman and Hall, New York (1993)

Fleiss, J.L.: Measuring nominal scale agreement among many raters. Psychological Bulletin 76(5), 378–382 (1971)

Hubert, L.: Kappa revisited. Psychological Bulletin 84(2), 289–297 (1977)

Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, New York (2011)

Kraemer, H.C.: Ramifications of a population model for κ as a coefficient of reliability. Psychometrika 44, 461–472 (1979)

Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004) ISBN : 0471210781

Light, R.L.: Measures of response agreement for qualitative data: some generalizations and alternatives. Psychological Bulletin 76, 365–377 (1971)

Miller, D.P., O'Shaughnessy, K.F., Wood, S.A., Castellino, R.A.: Gold standard and expert panels: a pulmonary nodule case study with challenges and solutions. In: Proceedings SPIE: Medical Imaging 2004: Image Perception, Observer Performance and Technology Assessment, vol. 5372, pp. 173–184 (2004)

Rao, C.R.: Linear Statistical Inference and its Applications, 2nd edn. Wiley, New York (2001)

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)

Salerno, S.M., Alguire, P.C., Waxman, S.W.: Competency in interpretation of 12-lead electrocardiograms: a summary and appraisal of published evidence. Annals of Internal Medicine 138, 751–760 (2003)

Schouten, H.J.A.: Measuring pairwise interobserver agreement when all subjects are judged by the same observers. Statistica Neerlandica 36, 45–61 (1982)

Scott, W.A.: Reliability of content analysis: The case of nominal scale coding. Public Opinion Q 19, 321–325 (1955)

Smith, R., Copas, A.J., Prince, M., George, B., Walker, A.S., Sadiq, S.T.: Poor sensitivity and consistency of microscopy in the diagnosis of low grade non-gonococcal urethrisis. Sexually Transmitted Infections 79, 487–490 (2003)

Snow, R., O'Connor, B., Jurafsky, D., Ng, A.Y.: Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In: EMNLP 2008: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 254–263. ACL (2008)

Soeken, K.L., Prescott, P.A.: Issues in the use of kappa to estimate reliability. Medical Care 24, 733–741 (1986)

Vanbelle, S., Albert, A.: Agreement between an isolated rater and a group of raters. Statistica Neerlandica 63(1), 82–100 (2009)

Warfield, S.K., Zou, K.H., Wells, W.M.: Simultaneous truth and performance level estimation (staple): an algorithm for the validation of image segmentation. IEEE Trans. Med. Imaging 23(7), 903–921 (2004)

Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., Movellan, J.: Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. Advances in Neural Information Processing Systems 22, 2035–2043 (2009)

Williams, G.W.: Comparing the joint agreement of several raters with another rater. Biometrics 32, 619–627 (1976)

Witten, I.H., Frank, E.: Weka 3: Data Mining Software in Java (2005), http://www.cs.waikato.ac.nz/ml/weka/

Yan, Y., Rosales, R., Fung, G., Schmidt, M., Hermosillo, G., Bogoni, L., Moy, L., Dy, J.: Modeling annotator expertise: Learning when everybody knows a bit of something. In: Proc. International Conference on Artificial Intelligence and Statistics (AISTATS). JMLR, vol. 9, pp. 932–939 (2010)

Compact Coding for Hyperplane Classifiers in Heterogeneous Environment

Hao Shao[1], Bin Tong[1], and Einoshin Suzuki[2]

[1] Graduate School of Systems Life Sciences, Kyushu University
[2] Department of Informatics, ISEE, Kyushu University

Abstract. Transfer learning techniques have witnessed a significant development in real applications where the knowledge from previous tasks are required to reduce the high cost of inquiring the labeled information for the target task. However, how to avoid *negative transfer* which happens due to different distributions of tasks in heterogeneous environment is still a open problem. In order to handle this kind of issue, we propose a Compact Coding method for Hyperplane Classifiers (CCHC) under a *two-level* framework in inductive transfer learning setting. Unlike traditional methods, we measure the similarities among tasks from the *macro level* perspective through minimum encoding. Particularly speaking, the degree of the similarity is represented by the relevant code length of the class boundary of each source task with respect to the target task. In addition, informative parts of the source tasks are adaptively selected in the *micro level* viewpoint to make the choice of the specific source task more accurate. Extensive experiments show the effectiveness of our algorithm in terms of the classification accuracy in both UCI and text data sets.

1 Introduction

Transfer learning [20] provides a solution of how to learn a target task in real applications where a large amount of auxiliary data from source domains are given. However, the negative transfer problem [13] tends to happen if the distributions of the source and the target domains are too dissimilar. A typical example is about language study, where learning English is the target task, and learning Japanese and French are regarded as the source tasks. It would be much effective for a learner to study English if he (she) is familiar with French. However, the learning process would be impeded if the learner only masters Japanese, because the lexical and semantic structure of Japanese is different from that of English. Negative transfer is more likely to happen once we underestimate the side effect resulting from the distribution differences of multiple source tasks [20], which is common in real applications.

To solve this problem, it is reasonable to measure the similarity between tasks and instances in heterogeneous environment [2,9,15,19] by considering the distribution dis(similarity) of different domains. For example, [15] proposed a dictionary-based compression dissimilarity measure between two instances in

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 207–222, 2011.

different domains for clustering. In this paper, we confine our focus on the classification problems using hyperplane lassifiers. Attempts to avoid negative transfer are made in transfer learning for classification by evaluating the distance between tasks [2,3] or distance between two instances [22,26,4] in different domains. For example, [2] tried to find a common representation among different groups of tasks which can be regarded as the transferred information. A novel kernel function is sophisticatedly designed in [4] by exploiting the Gaussian process model. Although this method is able to model the *negative* similarity between two instances, it can be only applied to one source task. We note that also, measuring the similarity between only tasks has a limited power, since the negative transfer would be induced when some instances are considered "too dissimilar" in one task compared to another, even if the the degree of the dissimilarity between the two tasks is very small. In addition, in the typical setting of the transfer learning, the labeled information is insufficient in the target task. It is difficult to get a reliable classifier by only considering the similarities between tasks [3].

Unlike the traditional methods above, we establish a different paradigm by proposing a method that consists of *macro level* and *micro level* evaluations, in order to spot relevant portions of the source tasks with the target task. Our fundamental idea is to measure the similarity in the *macro level* (task level) by compact coding which is inspired by the Minimum Description Length Principle (MDLP) [16]. The MDL Principle is built with a solid theoretical foundation that is suitable for model selection to avoid *overfitting*. It is successfully applied in the inductive transfer learning problem [10], but only confined to one source task and one target task. Put it another way, it is radical to consider that how to select informative knowledge from multiple source tasks can be conceptually regarded the same as the model selection. In our method, informative parts of the source tasks are adaptively selected in the *micro level* viewpoint to make the choice of the specific source task more accurate.

2 Problem Setting and Preliminaries for Encoding

In this paper, we deal with the inductive transfer learning problem where several source tasks are available. There is a task set $\mathbb{S}$ from the source domain which contains K tasks S_i $(i = 1, ..., K)$, one target training data set T and one test data set E from the target domain. Instances in S_i and T are labeled. All instances in S_i, T and E share the same attribute vector of m dimensions. From each data set, we obtain a hyperplane classifier $\mathbf{wx} = 0$, where $\mathbf{x} = (x_1, x_2, ..., x_m, 1)$. The weight vectors of hyperplanes for the source tasks are denoted by $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_K$, where $\mathbf{w}_i = (w_i^1, w_i^2, ..., w_i^m)$ and our objective is to find the weight vector $\mathbf{w}_t$ for the target task. In the transfer learning setting, the number of labeled instances in the target domain is assumed to be much smaller than that in the source domain. Although the labeled data is not sufficient to obtain an accurate hyperplane classifier on T, we can still draw a rough estimation using the limited labeled data, which is denoted by $\mathbf{v}_t$.

We provide here the preliminaries for MDLP, which may be viewed as a principle for avoiding overfitting, i.e., it is a means to balance the simplicity of a classifier and its goodness-of-fit to the data [7,11]. It states that the best classifier h_{best} is given as follows.

$$h_{\text{best}} = \arg\min_{h} \left(-\log P(h) - \log P(D|h)\right) \tag{1}$$

where h is a hypothesis on a data set D, and $P(h)$ and $P(D|h)$ represent the probability that h occurs and the conditional probability that D occurs given h, respectively. Consider the problem of encoding h as a binary string. According to the coding theory [6], the length of the code string for h using an optimally efficient code is $-\log P(h)$. Similarly, $-\log P(D|h)$ may be regarded as the length of the code string for D encoded with the help of h. In the MDLP for classification, these code lengths are calculated in a problem setting where the receiver has D except for the class labels of examples in D. The sender first sends h, then the class labels of examples in D with the help of h. The code used in this setting is a prefix code, in which no extension of a code word can itself be a code word. Intuitively, the hypothesis chosen by the MDLP coincides with the maximum a posteriori hypothesis.

The MDLP can be interpreted as assigning priors to theories based on a compact coding, i.e., $P(h)$ is defined by the coding method for calculating $-\log P(h)$. It is important that the coding method is efficient otherwise the assigned priors deviate from the philosophy of the MDLP.

We then consider a problem of encoding a binary string of length a which consists of b binary 1s and $(a-b)$ binary 0s under the framework of the sender and the receiver problem. An obvious method is to send the number b of binary 1s with the code length $\log(a+1)$ then specify the positions of binary 1s with the code length $\log \begin{pmatrix} a \\ b \end{pmatrix}$ [7,11]. We hereafter call this method a binary coefficient method and denote the required code length with $\Theta\,(a,b)$ as follows.

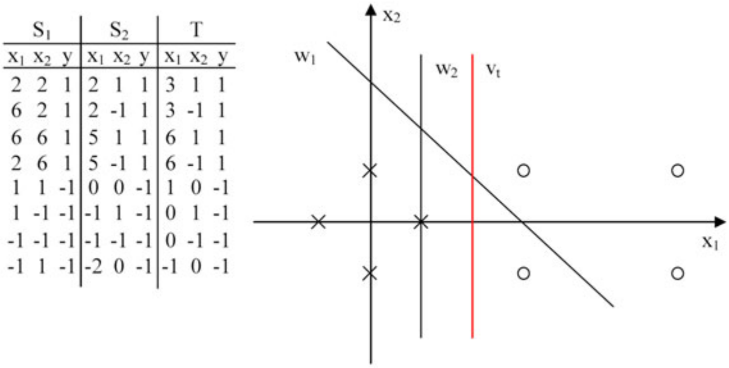
$$\Theta\,(a,b) \equiv \log(a+1) + \log \begin{pmatrix} a \\ b \end{pmatrix}$$

3 Compact Coding for Hyperplane Classifiers in Heterogeneous Environment

We provide in this section a detailed explanation of the coding scheme for our CCHC. Given a small number of labeled instances in the target domain, it is difficult to measure the similarity between a source task and the target task. Suppose there are two source tasks S_1, S_2 and one target training data set T, as shown in Fig. 1. The corresponding weight vectors of the hyperplanes are $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{v}_t$, $\mathbf{w}_1 = \{1, 1, -3\}$, $\mathbf{w}_2 = \{1, 0, -1\}$ and $\mathbf{v}_t = \{1, 0, -2\}$.

Intuitively, by applying $\mathbf{w}_1$ and $\mathbf{w}_2$ to T, we see from Fig. 1 that both $\mathbf{w}_1$ and $\mathbf{w}_2$ have 1 wrong prediction on T and it is difficult to tell which one is more similar to the target domain if no other information is provided.

To deal with such a problem, our main idea is to divide the classification task into a two-level evaluation: in the macro level, we evaluate the degrees of



S_1			S_2			T		
x_1	x_2	y	x_1	x_2	y	x_1	x_2	y
2	2	1	2	1	1	3	1	1
6	2	1	2	-1	1	3	-1	1
6	6	1	5	1	1	6	1	1
2	6	1	5	-1	1	6	-1	1
1	1	-1	0	0	-1	1	0	-1
1	-1	-1	-1	1	-1	0	1	-1
-1	-1	-1	-1	-1	-1	0	-1	-1
-1	1	-1	-2	0	-1	-1	0	-1

Fig. 1. An example of the three hyperplanes

similarity of the source tasks with the target task and in the micro level, we try to select useful parts of a relevant source task. In such a way, we intend to avoid negative transfer by adaptively selecting more relevant data from the source domain for training the classifier, and exclude those data which bring negative effects on the results.

Our algorithm includes two sub procedures:

1. ***Macro level:*** Sort S_i in descending order on the degrees of similarity with the target data set T.
2. ***Micro level:*** Divide the data set of the related source task into several components and select related parts to help training the classifier in the target domain.

3.1 Macro Level: Arrange Related Tasks

The objective of the macro level evaluation is to sort the source tasks based on the degrees of similarity with T. In the general MDLP framework, the model space contains candidates of models induced from the data, and the best model is obtained by minimizing the two-part code length. The model that fits the data more is assigned with a shorter code length. Consider $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_K$ are in the model space. Since each of the $\mathbf{w}_i$ is induced from the source task S_i, we call $P(\mathbf{w}_i|S_i)$ *a posteriori* probability for the source task S_i and a $\mathbf{w}_i$ that compresses the data well has a higher $P(\mathbf{w}_i|S_i)$. By substituting S_i with T, we infer that if $P(\mathbf{w}_i|T)$ is high, $\mathbf{w}_i$ is assumed to be a good hypothesis which can be induced from T. In such a case, we may say that S_i and T are similar. Therefore, $P(\mathbf{w}_i|T)$ could be used as a degree of similarity between S_i and T. Taking the negative logarithm function, the best model $\mathbf{w}^*$ to explain the data T is the one that minimizes the sum of the length, in bits, of the description of the model and the length of the description of the data when encoded with the help of the model. We could sort $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_K$ in descending order of $P(\mathbf{w}_i|T)$ as follows:

$$P(\mathbf{w}_i|T) \propto P(T|\mathbf{w}_i)P(\mathbf{w}_i)$$

Or in ascending order of the code length $(-\log P(\mathbf{w}_i) - \log P(T|\mathbf{w}_i))$.

Although the labeled information in T is not sufficient to obtain the optimal hyperplane in the target domain, $\mathbf{v}_t$ could be adopted to improve the performance as it is a rough estimation of the underlying hyperplane. We believe that it can be used to help sorting the source tasks by evaluating the degree of similarity between a source task S_i and T. In the learning process, it can be iteratively updated using transferred information from the source domain. Therefore, we take $\mathbf{v}_t$ into consideration to help coding the hyperplanes of the source tasks. We need to sort $\mathbf{w}_i$ in descending order of the posterior probability as follows:

$$P(\mathbf{w}_i|T, \mathbf{v}_t) \propto P(T, \mathbf{v}_t|\mathbf{w}_i)P(\mathbf{w}_i)$$

It can be further decomposed as:

$$\begin{aligned} P(\mathbf{w}_i|T, \mathbf{v}_t) &\propto P(T|\mathbf{w}_i)P(\mathbf{v}_t|\mathbf{w}_i)P(\mathbf{w}_i) \\ &\propto P(T|\mathbf{w}_i)P(\mathbf{w}_i|\mathbf{v}_t)P(\mathbf{v}_t) \\ &\propto P(T|\mathbf{w}_i)P(\mathbf{w}_i|\mathbf{v}_t) \end{aligned}$$

where $P(\mathbf{v}_t)$ is neglected in the third line because we could not settle a proper prior for $\mathbf{v}_t$ due to the small size of T, and it is of the same value when we compare different $\mathbf{w}_i$ given the specific $\mathbf{v}_t$.

By taking the negative log function,

$$L_i = -\log P(\mathbf{w}_i|\mathbf{v}_t) - \log P(T|\mathbf{w}_i) \tag{2}$$

The best $\mathbf{w}^*$ is the one that minimizes the two-part code length among all the models. Note that by compact coding, an optimal code assigns shorter codes to the model that suits the data most while others are given longer codes. This principle also coincides with the principle of minimum encoding. As an extension of the coding method that searches for the best hypothesis among all the candidates, we could use the code length as a similarity measure. Therefore, the degree of similarity between one source task S_i using the class boundary $\mathbf{w}_i$ in terms of the target task T can be represented by the code length as (2).

The first part is the complexity of one model given another and the second part can be treated as the negative log-likelihood of the data given the model. It is considered to be effective because the hyperplanes are concise summaries of the data, and in the classification tasks, it makes sense to use class boundaries in terms of T to measure the similarities between a source task and the target task.

Now let's consider the calculation of the first part by first introducing how to encode a real number x under the assumption that $x = \mu$ is most likely, where μ is also a real number. Let f be a continuous probability measure on the real numbers. Here we legitimately assume that f is a Gaussian, $f = 1/(2\pi\sigma^2)^{1/2}\exp(-(x-\mu)^2/2\sigma^2)$. Then we need to determine the value of σ.

The probability of x with precision ε is thus [25]:

$$P(x) = \int_{x-\frac{\varepsilon}{2}}^{x+\frac{\varepsilon}{2}} f(x)dx \approx \varepsilon f(x) \tag{3}$$

To obtain the variance σ, we assume that the probability of $x = \mu$ is q $(0 < q < 1)$, so we have

$$P(x = \mu) = \varepsilon \frac{1}{(2\pi\sigma^2)^{1/2}} = q \tag{4}$$

From (4), we can have that σ equals to $\varepsilon/(2\pi q^2)^{1/2}$, and in the calculation, we need to set two parameters ε and q. q ought to be high because in our coding scheme, x is more likely to take the value around μ. We set $\varepsilon = 0.01$ and $q = 0.8$ throughout the paper. Note that, a general precision is adequate for our similarity measure to distinguish different hyperplanes because the code lengths would be uniform for each hyperplane under the same precision.

Let $\Lambda(x, u)$ be the code length of sending x given μ,

$$\Lambda(x, u) = -\log\left(\varepsilon \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) = -\log\left(q e^{-\frac{\pi(x-\mu)^2 q^2}{\varepsilon}}\right) \tag{5}$$

Now we consider how to calculate $-\log P(\mathbf{w}_i)$, with the help of $\mathbf{v}_t$. $\mathbf{v}_t$ is taken into consideration because we want to use as much information as possible that can be inferred from the small number of labeled instances in the target task. Note that both $\mathbf{w}_i$ and $\mathbf{v}_t$ contain m entries, $\mathbf{w}_i = (w_i^1, w_i^2, ..., w_i^m)$ and $\mathbf{v}_t = (v_t^1, v_t^2, ..., v_t^m)$. Consider the sender and the receiver's problem, both the sender and the receiver have $\mathbf{v}_t$ in advance. To send $\mathbf{w}_i$, we assume that each entry in $\mathbf{w}_i$ is more likely to be approximate to the corresponding entry in $\mathbf{v}_t$. Therefore, the code length of $-\log P(\mathbf{w}_i)$ with the help of $\mathbf{v}_t$ should be

$$-\log P(\mathbf{w}_i|\mathbf{v}_i) = \sum_{j=1}^{m} \Lambda(w_i^j, v_t^j) \tag{6}$$

For the second part of (2), the class labels of all instances in T form a binary string, and thus it is convenient for us to send only the wrong predictions of $\mathbf{w}_i$ on this string. Let $\omega(\mathbf{w}_i, T)$ denote the number of misclassified examples on T,

$$-\log P(T|\mathbf{w}_i) := \Theta(|T|, \omega(\mathbf{w}_i, T)) \tag{7}$$

Combining equations (6) and (7), we have the code length L_i for $\mathbf{w}_i$ on T:

$$L_i = \sum_{j=1}^{m} \Lambda(w_i^j, v_t^j) + \Theta(|T|, \omega(\mathbf{w}_i, T)) \tag{8}$$

Then the source data sets could be arranged in ascending order on the code lengths with the target data set T. We have L_{min} as the shortest code length among the k code lengths.

For the example in Table 1, by calculating the code lengths for each of the hyperplane, we could obtain the following results:

$$L_1 = \Theta(|T|, \omega(\mathbf{w}_1, T)) + \sum_{j=1}^{3} \Lambda(w_1^j, v_t^j) = 587.31 bits$$

$$L_2 = \Theta(|T|, \omega(\mathbf{w}_2, T)) + \sum_{j=1}^{3} \Lambda(w_2^j, v_t^j) = 297.22 bits$$

Based on our similarity measure, $\mathbf{w}_2$ is more similar to $\mathbf{v}_t$ than $\mathbf{w}_1$. That means, the source data set S_2 is more similar to the target task. Therefore, we incline to select more information from S_2 in the micro level evaluation.

3.2 Micro Level Evaluation

In the macro level evaluation, the source data sets are sorted in descending order on the similarities with the target data set T, with indexes re-assigned from 1 to K. Suppose we now intend to transfer S_1 in the source domain to T, as some of the data in S_1 may not be useful, or sometimes harmful to the results, we need to adaptively select some parts of the data in the micro level evaluation.

A source task S_i is divided into n_s parts ($1 \leq n_s \leq |S_i|$). Our objective is to transfer only useful parts to T based on the coding method. Note that, if n_s equals to 1, all the instances in S_i are transferred and if n_s equals to $|S_i|$, it coincides with the instance-based transfer methods which evaluate all instances in the data sets.

To divide the source data sets, a simple clustering method such as k-means can be adopted in our framework. When we select one related source task S_i, we first perform clustering on this data set, which generates n_s clusters, namely $S_i^1, S_i^2, ..., S_i^{n_s}$. Each cluster is then regarded as a data set and is evaluated in the same way as the macro level by compact coding, with a code length assigned to evaluate the degree of similarity to the target task T.

3.3 The Transfer Learning Algorithm

We provide our CCHC algorithm as follows:

```
Algorithm CCHC
  for $i = 1$ to $K$
    calculate $L_i$ for each $S_i$ by (8), obtain $L_{min}$
    sort $S_i$ based on the ascending order of $L_i$
  $TR = \emptyset$
  for $j = 1$ to $K$
    perform clustering on $S_j$, obtain $S_j^t$ ($t = 1,...,n_s$)
    calculate $l_t$ for each $S_j^t$ by (8)
    sort $S_j^t$ based on the ascending order of $l_t$
    for $t = 1$ to $n_s$
```

```
        $TR = TR \cup S_j^t$ with the shortest $l_t$
        perform classification by SVM on $TR$ and obtain $\mathbf{w}'$
        calculate $L' = -\log P(\mathbf{w}'|\mathbf{v}_t) - \log P(T|\mathbf{w}')$
        if $L' < L_{min}$
            $L_{min} = L'$
            $\mathbf{v}_t = \mathbf{w}'$
            $S_j = S_j - S_j^t$
        else break
    $\mathbf{w}_t = \mathbf{v}_t$
    output $\mathbf{w}_t$
```

where TR denotes the training set for the classification task. l_t represents the code length for each part of the source task in the micro level evaluation.

4 Experiments

4.1 Experimental Setting

We perform experiments on the data sets from the UCI repository[1] and the Text data sets. The three data sets in the UCI repository used in the experiments are *mushroom*, *splice* and *krvskp*. We adopted a pre-processing method [22,26] on them to fit the transfer learning scenario. For the Text data sets, we choose 20NewsGroup data sets[2].

The *mushroom* data set has 8124 examples with 22 attributes in each example and one binary class label. The *splice* data set has 3190 examples with 60 attributes in each example and one binary class label. The *mushroom* data set is split into the source task and the target task based on the attribute *stalk-shape*, the source task contains examples whose *stalk-shape* are *tapering* and the examples in the target task have *stalk-shape* of *enlarging*. The *splice* data set is divided into two based on the first attribute. If the first attribute value of an example is “A” or “G”, it is added into the target task, otherwise it is added into the source task. [22] and [26] show that the splitting method could ensure different distributions between the source and the target tasks. The *krvskp* data set has 3196 examples with 36 attributes in each example and one binary class label. Since it is not included in [26], we adopt the same strategy to split the data into two. The data set is divided based on the eleventh attribute as it shows a result similar to the *splice* data set. The source and the target tasks contain examples with the attribute value f and t, respectively.

The number of examples in the source task for the *mushroom* data, the *splice* data and the *krvskp* data is set to be 1000 which is the same as in [26]. We investigated the influence of the number of instances in the target data set, and the noise level in the target data set. The noise is added by reversing the correct class labels of the examples in the training data sets.

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

[2] http://people.csail.mit.edu/jrennie/20Newsgroups

We follow the splitting strategy on 20Newsgroups data sets as [22]. Three data sets are chosen which are *rec* vs *talk*, *rec* vs *sci* and *sci* vs *talk*. For example, in *rec* vs *talk* data set, all the positive instances are from the category *rec*, while negative ones are from the category *talk*. The instances in the source domain and the target domain are selected based on the subcategories. In the experiments, each of the target tasks in the three data sets are chosen as the single target task, and the training data in the three data sets are all chosen as the source tasks. In such a way, $\mathbb{S}$ contains 3 different tasks as S_1, S_2 and S_3 and we test our algorithm in this transfer learning setting. Note that, 20Newsgroups data is regarded to be class-balanced. The challenging issues such as handling class-imbalanced data will be considered as one of our future works.

Our CCHC is compared with the COITL [26] and TrAdaBoost [22], which are two state-of-the-art instance-based methods in transfer learning, Active Transfer (AT) [23] which is a feature-based method, and k-NN with $k = 3$ as well as the basic SVM. We follow the values of the parameters in the original papers. The source data sets are merged into one data set for these algorithms. k-means clustering method was chosen as the clustering method in our micro level evaluation. After carefully testing the parameters, for *mushroom*, *splice* and *krvskp* data sets, k is set to be 4 and for the remaining data sets, k=2. All the experiments are repeated ten times and we report the average results. The hyperplanes are obtained by C-SVC with polynomial kernel [5], which are considered effective to data with a large number of features and without class noise.

For the UCI data sets, each of *mushroom*, *splice* and *krvskp* has one source task and one target task, CCHC is thus used only in the micro level to select useful parts. For the Text data sets, both the macro level and the micro level evaluations are examined. We mainly test two factors in the experiments. One is the different number of instances in the target task. We set $|T|$ equals to 50 and 100, respectively. The other is the noise level in the target task from 0% to 15%. For example, when $|T|$=50 and the noise level is 15%, there are only a small number of instances correctly labeled and this number can be regarded as "few". We also check the number of source tasks and the number of clusters in each task that are transferred to the target task.

4.2 Experimental Results

Table 1, 2 and 3 provide the results on the *mushroom* data set, the *krvskp* data set and the *splice* data set, respectively. As the general tendency, the error rates become larger with the noise level increases from 0% to 15%. Also, given more labeled instances in the target domain, such as $|T| = 100$, the results are obviously better. It can be observed from the tables that our method outperforms the other methods in most conditions. For the *mushroom* data set in Table 1, however, our method is outperformed by others in a small number of conditions such as in 15% noise level while $|T| = 50$. The possible reason is that, *mushroom* data sets are well organized and it is simple to find the underlying hypotheses even with a few labeled instances. The improvements by transferring information from the source domain to the target domain are not as promising as expected.

Table 1. Results on *mushrom* data set

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$\|T\|$=50	SVM	0.087	0.12	0.146	0.173	0.19	0.207
	CCHC	**0.079**	**0.106**	**0.129**	**0.127**	**0.156**	0.171
	TrAdaBoost	0.158	0.159	0.173	0.191	0.168	0.195
	KNN	0.117	0.125	0.153	0.147	0.167	0.163
	COITL	0.132	0.144	0.146	0.161	0.168	0.159
	AT	0.156	0.196	0.177	0.16	0.185	**0.142**
$\|T\|$=100	SVM	0.067	0.052	0.111	0.129	0.167	0.198
	CCHC	**0.058**	**0.05**	**0.081**	**0.116**	0.162	0.179
	TrAdaBoost	0.145	0.143	0.158	0.178	0.167	0.166
	KNN	0.081	0.084	0.104	0.12	0.147	0.159
	COITL	0.103	0.08	0.087	0.121	**0.11**	**0.112**
	AT	0.2	0.189	0.177	0.199	0.184	0.178

Table 2. Results on *krvskp* data set

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$\|T\|$=50	SVM	0.15	0.22	0.192	0.167	0.256	0.282
	CCHC	**0.095**	**0.128**	**0.126**	0.143	**0.153**	**0.18**
	TrAdaBoost	0.514	0.504	0.504	0.489	0.491	0.493
	KNN	0.25	0.277	0.328	0.303	0.297	0.311
	COITL	0.246	0.283	0.33	0.288	0.285	0.313
	AT	0.18	0.192	0.144	**0.13**	0.18	0.19
$\|T\|$=100	SVM	0.11	0.099	0.171	0.166	0.243	0.207
	CCHC	**0.066**	**0.085**	**0.115**	**0.132**	0.167	0.169
	TrAdaBoost	0.509	0.505	0.496	0.484	0.497	0.478
	KNN	0.221	0.215	0.224	0.242	0.24	0.276
	COITL	0.24	0.219	0.229	0.24	0.233	0.276
	AT	0.11	0.133	0.122	0.135	**0.127**	**0.138**

In Table 2 of *krvskp*, our method outperforms other methods in most conditions. In Table 3 of *splice*, our method is the best one among all the methods even with 15% noise. And the improvements are nearly 10% compared to those of the state-of-the-arts methods. From the experiments, we observed that the quality of $\mathbf{v}_t$ tends to fluctuate when the noise is high. In *splice* data sets with 60 attributes, this impact becomes less serious and CCHC could gain much help from the abundant information in the source tasks to improve the hyperplane in the target task.

We also evaluate the number of source tasks and the number of clusters in the source tasks that are transferred in Table 4, where S denotes the source task for the *mushroom*, *krvskp* and *splice* data sets. The values in the table represent

Table 3. Results on *splice* data set

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$\|T\|$=50	SVM	0.302	0.321	0.354	0.368	0.391	0.346
	CCHC	**0.18**	**0.207**	**0.212**	**0.214**	**0.216**	**0.222**
	TrAdaBoost	0.343	0.296	0.318	0.323	0.314	0.39
	KNN	0.375	0.354	0.347	0.379	0.398	0.415
	COITL	0.385	0.401	0.388	0.422	0.394	0.436
	AT	0.468	0.47	0.441	0.469	0.48	0.478
$\|T\|$=100	SVM	0.232	0.23	0.276	0.293	0.309	0.322
	CCHC	**0.18**	**0.182**	**0.184**	**0.199**	**0.209**	**0.203**
	TrAdaBoost	0.234	0.3	0.302	0.267	0.294	0.299
	KNN	0.331	0.349	0.352	0.377	0.367	0.396
	COITL	0.319	0.339	0.327	0.362	0.367	0.374
	AT	0.472	0.458	0.474	0.484	0.465	0.477

Table 4. Transferred components of the source tasks in UCI data sets

			Percentage of noise on T					
			0%	3%	6%	9%	12%	15%
mushroom	$\|T\| = 50$	S	2	2	1	1	2	1
	$\|T\| = 100$	S	2	1	2	2	2	2
krvskp	$\|T\| = 50$	S	1	2	3	3	2	2
	$\|T\| = 100$	S	1	2	3	2	2	3
splice	$\|T\| = 50$	S	2	3	1	2	3	3
	$\|T\| = 100$	S	2	2	2	3	3	3

the number of clusters in each source task that are used in the learning process. As shown in Table 4, in the two data sets *krvskp* and *splice*, the components of the source tasks are transferred as much as possible. The reason is that in the two data sets, the number of attributes is more than that in the *mushroom* data set, and the labeled instances in the target training task are not adequate to obtain good hyperplanes. Therefore, our method tries to find relevant information from the source tasks as much as possible.

Table 5, Table 6 and Table 7 provide the results on *rec* vs *talk*, *rec* vs *sci*, and *sci* vs *talk*, respectively, with $|T|$ equals to 50 and 100. It can be seen from the three tables that our method is obviously better than other methods, even under noise conditions. However, CCHC is sometimes outperformed by COITL or TrAdaBoost in a small number of situations while $|T| = 100$. We ascribe the results to the reason that when the number of labeled instances increases, other methods also could find good hyperplanes. But under the circumstances when $|T| = 50$, our method is always the best one. It is a proof of the robustness of our method given only a few labeled instances in the target domain. We also

Table 5. Results on *rec* vs *talk* as the target task

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$\|T\|$=50	SVM	0.154	0.174	0.189	0.228	0.227	0.23
	CCHC	**0.122**	**0.144**	**0.167**	**0.196**	**0.21**	**0.223**
	TrAdaBoost	0.236	0.234	0.275	0.309	0.32	0.321
	KNN	0.207	0.255	0.237	0.256	0.244	0.305
	COITL	0.206	0.255	0.229	0.25	0.241	0.294
	AT	0.472	0.364	0.38	0.388	0.488	0.49
$\|T\|$=100	SVM	0.088	0.166	0.177	0.269	0.285	0.295
	CCHC	**0.084**	**0.154**	**0.151**	**0.186**	**0.191**	**0.248**
	TrAdaBoost	0.265	0.283	0.338	0.348	0.355	0.338
	KNN	0.23	0.248	0.267	0.277	0.281	0.283
	COITL	0.224	0.245	0.257	0.267	0.278	0.283
	AT	0.406	0.363	0.492	0.477	0.315	0.517

Table 6. Results on *rec* vs *sci* as the target task

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$\|T\|$=50	SVM	0.163	0.177	0.187	0.181	0.223	0.24
	CCHC	**0.159**	**0.167**	**0.179**	**0.18**	**0.221**	**0.227**
	TrAdaBoost	0.266	0.289	0.292	0.353	0.351	0.375
	KNN	0.245	0.224	0.255	0.246	0.261	0.3
	COITL	0.243	0.236	0.259	0.255	0.268	0.311
	AT	0.41	0.37	0.44	0.325	0.354	0.419
$\|T\|$=100	SVM	0.139	0.152	0.153	0.174	0.21	0.213
	CCHC	**0.126**	**0.128**	**0.15**	**0.168**	**0.208**	**0.21**
	TrAdaBoost	0.24	0.23	0.24	0.228	0.267	0.284
	KNN	0.216	0.185	0.192	0.182	0.212	0.231
	COITL	0.206	0.2	0.194	0.183	0.22	0.231
	AT	0.433	0.316	0.399	0.359	0.315	0.438

notice that, when the *sci* vs *talk* data set is used as the target task, the error rates for all the methods are slightly higher than that of the other two data sets. The possible reason is that the discrepancy of distributions between the source domain and the target domain is large.

In Table 8, we report the number of source tasks and the number of clusters in each source task used in the micro level stage, where S_1, S_2 and S_3 denote the source task from *rec* vs *talk*, *rec* vs *sci* and *sci* vs *talk*, respectively. The integer values in the table represent the numbers of clusters in each source task that are used in the micro level. It can be observed that, our method is able to choose relevant parts from relevant tasks. For example, when *rec* vs *talk* is used as the target task, by examining the learning procedure of the algorithm,

Table 7. Results on *sci* vs *talk* as the target task

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$\|T\|$=50	SVM	0.183	0.195	0.223	0.24	0.307	0.364
	CCHC	**0.174**	**0.189**	**0.205**	**0.232**	**0.244**	**0.272**
	TrAdaBoost	0.301	0.282	0.317	0.372	0.372	0.409
	KNN	0.285	0.305	0.301	0.327	0.307	0.385
	COITL	0.286	0.303	0.318	0.327	0.31	0.385
	AT	0.368	0.425	0.446	0.382	0.366	0.371
$\|T\|$=100	SVM	0.221	0.235	0.239	0.264	0.329	0.342
	CCHC	**0.167**	**0.174**	**0.172**	0.21	0.232	0.254
	TrAdaBoost	0.195	0.219	0.255	0.244	0.213	**0.23**
	KNN	0.184	0.185	0.196	0.194	0.194	0.233
	COITL	0.172	0.184	0.188	**0.188**	**0.192**	0.234
	AT	0.352	0.399	0.323	0.327	0.418	0.362

Table 8. Transferred components of the source tasks in 20Newsgroup data sets

			Percentage of noise on T					
			0%	3%	6%	9%	12%	15%
rec vs *talk* as T	$\|T\| = 50$	S_1	0	0	0	1	0	0
		S_2	1	1	1	1	1	0
		S_3	1	1	1	1	1	1
	$\|T\| = 100$	S_1	1	0	0	0	0	0
		S_2	1	1	1	1	1	1
		S_3	1	1	1	1	1	1
rec vs *sci* as T	$\|T\| = 50$	S_1	1	1	0	1	1	1
		S_2	0	0	0	0	0	0
		S_3	1	1	1	1	1	0
	$\|T\| = 100$	S_1	1	1	1	1	1	1
		S_2	0	0	0	0	1	0
		S_3	0	0	1	1	0	1
sci vs *talk* as T	$\|T\| = 50$	S_1	1	1	1	1	1	1
		S_2	0	0	0	0	0	0
		S_3	1	1	1	0	0	1
	$\|T\| = 100$	S_1	1	1	0	1	1	1
		S_2	0	0	0	0	0	0
		S_3	1	1	1	1	1	1

we found that in the macro level, our method arrange the source tasks as S_3, S_2 and S_1, thus CCHC is more likely to choose the clusters in S_3 and S_2, while the third source task is regarded as to bring negative effects on the results. It can be a proof of the effectiveness of our method that can adaptively choose relevant information from the source domain in order to avoid negative transfer.

5 Related Work

In this section, we review research works similar to ours including different methods for inductive transfer learning, and related works on negative transfer. Our CCHC belongs to the supervised inductive transfer learning where both the source and the target tasks contain labeled data [18,3,19]. Existing methods for inductive transfer learning may be classified into instance-based approaches [22,26] and feature-based approaches [1,21,23,24]. For instanced-based methods, each instance in the source task is evaluated before being added to the target task to help the classification process in the target task. Both [22] and [26] adopted a re-weighting method to judge the impact of examples in the source task in the learning of the target task. The feature-based methods try to find a subspace spanned by relevant features as common knowledge to help to improve the classification results on the target task. However, most of them are not parameter-free and sensitive to noise. [1] proposed a method to learn a low-dimensional representation for multiple tasks. [24] provided a spectral-based solution to transfer the eigenspace where the process is adjusted by the KL divergence. [21] tried to find a kernel given a set of predefined kernels. Our method also belongs to the feature-based transfer. For feature selection, the MDLP based method was proposed in [17] to learn prior knowledge over the features.

To tackle with the problem of negative transfer, current works mainly focus on finding the similarities among tasks or instances such as [2,4,23,22,26,9]. [22] extended the AdaBoost algorithm which aims at improving the accuracy of a weak learner by adjusting the weights of the instances in the training sets. Their TrAdaBoost algorithm could evaluate the instances from a large amount of data in the source domain and assign weights based on the the similarities to the target task in order to boost the accuracy of the classifier. [26] also proposed a semi-supervised learning method by extending the co-training method which deals with the same problem as [22]. Instances that can be put into the target task are obtained by re-weighting those in the source task. [2] tried to find a common representation among different groups of tasks which can be regarded as the transferred information. However, the method was restricted to linear classification functions and the instances in the target task are not considered to be much less than that in the source tasks. The basic Probabilistic Latent Semantic Analysis (PLSA) is extended in [9], in order to simultaneously capture both the domain distinctions and commonality among multiple domains. In [4], a new kernel function was designed by exploiting the Gaussian process to evaluate the negative similarity between two instances, but it is designed for only one single source task and without taking multiple source tasks into the framework. An active learning approach ERS [14] (Error Reduction Sampling) is integrated into the transfer learning [23] with a heuristic similarity function. It is pointed out by the author that experts are heavily relied on, as the possibility to query an expert is set to be higher than 50%. Moreover, the degree of reliability of the oracles is not fully taken into consideration. Our work relates to the feature-based transfer learning, and a two-level evaluation method was proposed, in which the similarities among both the tasks and the components of tasks are considered.

6 Conclusion

This paper proposed a compact coding algorithm CCHC for inductive transfer learning to solve the negative transfer problem. The coding scheme is inspired by MDLP and a two-level evaluation is adopted to adaptively select useful parts in the source domain, and meantime to neglect those parts which will result in negative effects from further consideration. By incorporating a simple classifier such as SVM, given a few labeled examples, our method performs well on the real data sets, and outperforms other state-of-the-art methods in most conditions.

Acknowledgment. This work was partially supported by the grant-in-aid for scientific research on fundamental research (B) 21300053 from the Japanese Ministry of Education, Culture, Sports, Science and Technology, and the Strategic International Cooperative Program funded by the Japan Science and Technology Agency (JST).

References

1. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-Task Feature Learning. In: Proc. 19th NIPS, pp. 41–48 (2007)
2. Argyriou, A., Maurer, A., Pontil, M.: An Algorithm for Transfer Learning in a Heterogeneous Environment. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 71–85. Springer, Heidelberg (2008)
3. Bakker, B., Heskes, T.: Task Clustering and Gating for Bayesian Multitask Learning. Journal of Machine Learning Research 4, 83–99 (2003)
4. Cao, B., Pan, S.J., Yang, Q.: Adaptive Transfer Learning. In: AAAI 2010 (2010)
5. Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines(2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
6. Shannon, C.: A Mathematical Theory of Communication. Bell System Technical Journal 27, 379–423, 623–656 (1948)
7. Wallace, C., Patrick, J.: Coding Decision Trees. Machine Learning 11(1), 7–22 (1993)
8. Keogh, E., Lonardi, S., Ratanamahatana, C.A.: Towards Parameter-Free Data Mining. In: KDD 2004, pp. 206–215 (2004)
9. Zhuang, F.Z., Luo, P., Shen, Z.Y., He, Q., Xiong, Y.H., Shi, Z.Z., Xiong, H.: Collaborative Dual-PLSA: Mining Distinction and Commonality across Multiple Domains for Text Classification. In: CIKM 2010, Toronto, Canada (Octorber 2010)
10. Shao, H., Suzuki, E.: Feature-based Inductive Transfer Learning through Minimum Encoding. In: SDM 2011, Phoenix/Mesa, Arizona (April 2011)
11. Quinlan, J.R., Rivest, R.L.: Inferring Decision Trees Using the Minimum Description Length Principle. Information and Computation 80(3), 227–248 (1989)
12. Borgwardt, K.M., Gretton, A.: Integrating Structured Biological Data by Kernel Maximum Mean Discrepancy. Bioniformatics, 1–9 (2006)
13. Rosenstein, M.T., Marx, Z., Kaelbling, L.P.: To Transfer or Not To Transfer. In: NIPS 2005 Workshop on Transfer Learning (2005)
14. Roy, N., McCallum, A.: Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In: ICML 2001, pp. 441–448 (2001)

15. Thach, N.H., Shao, H., Tong, B., Suzuki, E.: A Compression-based Dissimilarity Measure for Multi-task Clustering. In: Proc. Nineteenth International Symposium on Methodologies for Intelligent Systems, Warsaw (June 2011)
16. Grünwald, P.D.: The Minimum Description Length Principle. MIT Press, Cambridge (2007)
17. Dhillon, P.S., Ungar, L.: Transfer Learning, Feature Selection and Word Sense Disambiguation. In: NLP/CL ACL-IJCNLP, Singapore (August 2009)
18. Caruana, R.: Multitask Learning. Machine Learning 28(1), 41–75 (1997)
19. David, S.B., Schuller, R.: Exploiting Task Relatedness for Multiple Task Learning. In: COLT 2003, pp. 825–830 (2003)
20. Pan, S.J., Yang, Q.: A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2008)
21. Rückert, U., Kramer, S.: Kernel-based Inductive Transfer. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 220–233. Springer, Heidelberg (2008)
22. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for Transfer Learning. In: ICML 2007, pp. 193–200 (2007)
23. Shi, X., Fan, W., Ren, J.: Actively Transfer Domain Knowledge. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 342–357. Springer, Heidelberg (2008)
24. Shi, X., Fan, W., Yang, Q., Ren, J.: Relaxed Transfer of Different Classes via Spectral Partition. In: ECML/PKDD, vol. (2), pp. 366–381 (2009)
25. Ke, Y.: Inferring Informed Clustering Problems with Minimum Description Length principle. Doctoral dissertation, State University of New York at Albany (2007)
26. Shi, Y., Lan, Z.Z., Liu, W., Bi, W.: Extended Semi-supervised Learning Methods for Inductive Transfer Learning. In: ICDM 2009, pp. 483–492 (2009)

Multi-label Ensemble Learning

Chuan Shi[1], Xiangnan Kong[2], Philip S. Yu[2], and Bai Wang[1]

[1] School of Computer
Beijing University of Posts and Telecommunications, Beijing, China
[2] Department of Computer Science
University of Illinois at Chicago, IL, USA
{shichuan,wangbai}@bupt.edu.cn, {xkong4,psyu}@uic.edu

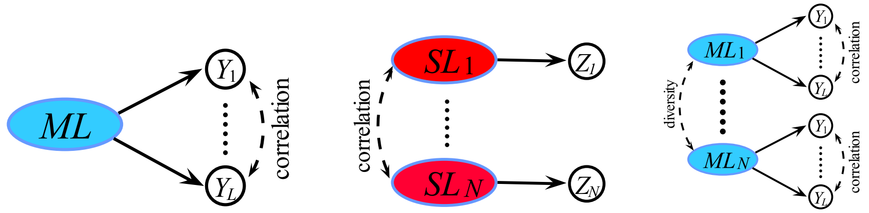
Abstract. Multi-label learning aims at predicting potentially multiple labels for a given instance. Conventional multi-label learning approaches focus on exploiting the label correlations to improve the accuracy of the learner by building an individual multi-label learner or a combined learner based upon a group of single-label learners. However, the generalization ability of such individual learner can be weak. It is well known that ensemble learning can effectively improve the generalization ability of learning systems by constructing multiple base learners and the performance of an ensemble is related to the both accuracy and diversity of base learners. In this paper, we study the problem of multi-label ensemble learning. Specifically, we aim at improving the generalization ability of multi-label learning systems by constructing a *group* of *multi-label base learners* which are both *accurate* and *diverse*. We propose a novel solution, called EnML, to effectively augment the accuracy as well as the diversity of multi-label base learners. In detail, we design two objective functions to evaluate the accuracy and diversity of multi-label base learners, respectively, and EnML simultaneously optimizes these two objectives with an evolutionary multi-objective optimization method. Experiments on real-world multi-label learning tasks validate the effectiveness of our approach against other well-established methods.

Keywords: Multi-label learning, ensemble learning, multi-objective optimization, negative correlation learning.

1 Introduction

Traditional supervised learning works on the single-label scenario, i.e. each instance is associated with one single label within a finite set of labels. However, in many real-world applications, one instance can be associated with multiple labels simultaneously. For example, in text categorization, each document may belong to several topics [20]; in bioinformatics, each gene may be associated with a number of functional classes [6]. This kind of problem is called multi-label learning, which corresponds to the classification problem of classifying each instance with a *set* of labels within the space of possible label sets. Multi-label learning has been drawing increasing attentions from the machine learning and data mining communities in the past decade [5,13,25].

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 223–239, 2011.



(a) Individual multi-label learner

(b) Multiple single-label learners

(c) Multi-label ensemble learning

Fig. 1. Comparison of three strategies of constructing multi-label learning system. *SL* and *ML* represent the single-label and multi-label learner, respectively. *Y* and *Z* represent the single and atomic label, respectively.

The multi-label learning faces a major challenge that the number of possible label combinations grows exponentially. Conventional multi-label learning approaches focus on exploiting the label correlations to improve the accuracy of individual multi-label learner [5,13,15,17,25]. These approaches can be roughly characterized into the following two categories based on the strategy of constructing the learning system: (1) Multi-label learning approaches based upon individual multi-label learner (shown in Figure 1(a)). In this type of approaches, a multi-label learner is constructed to make predictions on all labels. The label correlations are exploited in the structure or learning process of the multi-label learner, such as the neural network structure in ML-RBF [21] and the Bayesian learning in LEAD [25]. (2) Multi-label learning approaches based upon a group of single-label learners (shown in Figure 1(b)), such as EPS [14] and RAKEL [17]. Ensemble learning is used to construct such a group of single-label base learners. Each base learner in the ensemble is constructed to make a prediction on a single label or atomic label (i.e. treating each label subset as a class label). Then those base learners are combined as one multi-label learner to make predictions on all labels. The label correlations are usually exploited among these single-label base learners.

Generally, conventional multi-label learning approaches focus on building one individual multi-label learner. However, the generalization ability of one individual learner can be weak. It is well-known that ensemble learning can improve the generalization ability of a learning system and reduce the overfitting risk by constructing multiple base learners in the single-label setting. In the case of multi-label learning, if we ensemble a group of multi-label base learners to make predictions on all labels, the generalization ability of the multi-label learning system can be significantly improved. This is called the multi-label ensemble learning problem (shown in Figure 1(c)). Since the generalization error of an ensemble is related to the average generalization error of the base learners as well as diversity among the base learners [10], the aim of multi-label ensemble learning is to build a *group* of *multi-label base learners* which are not only *accurate* but also *diverse*. Note that, different from previous ensemble methods for multi-label learning which combine a group of single-label base learners into one multi-label learner, the base learners in the multi-label ensemble learning are the multi-label learners, instead of the single-label learners.

Despite its value and significance, the multi-label ensemble learning has rarely been studied in this context so far. It is challenging to generate a set of accurate and diverse multi-label base learners in the multi-label scenario. The major research challenges are as follows: (1) Conventional ensemble learning approaches usually focus on single-label learning problems. When it is applied to multi-label learning problems, one of the difficulties is the accuracy evaluation of multi-label base learners, which needs to consider the correlations among different labels. (2) In multi-label scenario, it is also difficult to evaluate the diversity of multi-label base learners, since the output of the base learners is a label set (vector), instead of a single label (scale number). (3) It is far more complex when considering the accuracy and diversity of multi-label base learners simultaneously. We need to consider how to balance the trade-off between these two aspects.

In this paper, we first study the problem of multi-label ensemble learning and propose a novel solution, named EnML. Different from conventional multi-label learning approaches, EnML builds a group of accurate and diverse multi-label base learners. First, we propose two novel evaluation objectives to effectively depict the accuracy and diversity of multi-label base learners, respectively. Inspired by the Hilbert-Schmidt Independence Criterion (HSIC) [8], *ML-HSIC* is proposed to evaluate the accuracy of base learners while considering the label correlations in full order. Enlightened by the Negative Correlation Learning (NCL) [11,12], *ML-NCL* is proposed to characterize the diversity of base learners. In order to balance the trade-off between these two objectives for the generalization ability of the ensemble, we then propose a novel evolutionary multi-objective algorithm to search the optimal trade-off among the different objectives. Extensive experiments on the different types of multi-label datasets show that EnML significantly outperforms other popular multi-label learning approaches.

2 Related Work

In order to improve the generalization ability of multi-label learner system, conventional approaches focus on building an accurate multi-label learner by exploiting the label correlations. According to strategies of building learner, conventional multi-label learning approaches can be roughly classified into following two categories. (1) The first type of approaches build an individual multi-label learner to make predictions on all labels. The multi-label learner uses different methods to exploit the label correlations, such as learner structure, optimized criterion, and learning algorithm. For example, the neural network structures in ML-RBF [21] and BP-MLL [23] mix the label relations, the *ranking loss* criterion [6,25] considers the second order correlation of labels, and the Bayesian learning in LEAD [25] learns the label dependency. EnML is different from this type of approaches in building a *group* of multi-label learners. (2) The second type of approaches build a set of single-label base learners. In these approaches, each single-label base learner is built to make a prediction on a single label or atomic label, and then these base learners are combined as a multi-label learner. The label correlations can be exploited among these base learners. Ensemble learning is usually used to build such a set of single-label base learners [14,15,16,17]. For example, RAKEL [17] trains each single-label base learner for the prediction of each element in the powerset of label set, and the single-label base learner in EPS [14] is built for a pruning label subset. Different from these ensemble methods for multi-label learning, the base learners in EnML are the *multi-label learners*.

Ensemble of multiple learners has attracted a lot of research interest in the machine learning community since it is considered as a good approach to improve the generalization ability [2]. Most ensemble learning algorithms train the individual learner independently or sequentially, so the advantages of interaction and cooperation among the individual learners are not effectively exploited. However, Liu and Yao [11,12] have shown that the cooperation with ensemble members is useful for obtaining better ensembles. Negative Correlation Learning (NCL) [3,11,12] introduces a correlation penalty term into the error function of each individual base learner in the ensemble so that the learners are as different as possible on the training error. NCL emphasizes the interaction and cooperation among individual base learners in the ensemble and has performed well on a number of empirical applications. However, the conventional NCL focuses on single-label learning, and has never been applied in multi-label learning so far.

3 The EnML Method

Let $\chi = \mathbb{R}^d$ be the d-dimensional input space and $\mathcal{L} = \{1, 2, \cdots, L\}$ be the finite set of L possible labels. Given a multi-label training set $\mathcal{D} = \{(\mathbf{x}_i, Y_i) | 1 \leq i \leq m\}$, where $\mathbf{x}_i \in \chi$ is the i-th instance and $Y_i \subseteq \mathcal{L}$ is the label set associated with $\mathbf{x}_i$. The task of multi-label learning is to learn a multi-label learner $h : \chi \rightarrow 2^{\mathcal{L}}$ from $\mathcal{D}$ which predicts a set of labels for each unseen instance.

As we all know, the ensemble can improve the generalization ability of learning systems by constructing multiple base learners, and the ensemble members should be accurate and diverse [10]. In multi-label ensemble learning, we aim at building such an ensemble, in which each multi-label base learner has good classification performances while these base learners are as diverse as possible. To do so, we propose a multi-objective optimization based solution. Concretely, we proposes two novel criteria, *ML-HSIC* and *ML-NCL*, to evaluate the accuracy and diversity of multi-label base learners, respectively. An evolutionary multi-objective optimization algorithm is then designed to train a set of multi-label base learners which are diverse and all optimal in these two proposed criteria.

3.1 Measure Criteria

ML-HSIC. Many criteria have been proposed to evaluate performances of multi-label learning, such as *hamming loss* [21] and *ranking loss* [23]. These criteria can be used as the accuracy evaluation. However, they fail to directly address the correlations between different labels. An ideal criterion needs to be able to evaluate the accuracy of learners while considering the label correlations.

The accuracy of a multi-label learner h can be considered as the similarity of the true label set $TL = \{Y_1, \cdots, Y_m\}$ and the predicted label set by h on the training data $\mathcal{D}$, $PL = \{h(\mathbf{x}_1), \cdots, h(\mathbf{x}_m)\}$. Furthermore, the similarity can be evaluated with the dependence between PL and TL. That is, the higher dependence between PL and TL, the more similar they are. Many methods can be used to characterize the dependence. In this paper, we derive an evaluation criterion for multi-label learning based upon a dependence evaluation method named Hilbert-Schmidt Independence Criterion (HSIC) [8]. By deriving from the definition of HSIC, we define the accuracy of a learner h as follows:

$$\textit{ML-HSIC}(h) = tr(PHQH) \tag{1}$$

where $tr(\cdot)$ is the trace of a matrix and $H = [h_{ij}]_{m\times m}$, $h_{ij} = \delta_{ij} - 1/m$, and δ_{ij} is the indicator function which takes 1 when $i = j$ and 0 otherwise. $P = [p_{ij}]_{m\times m}$ denotes the label kernel matrix based on the true label set TL with the kernel function $p(Y_i, Y_j) = \langle\phi(Y_i), \phi(Y_j)\rangle$. $Q = [q_{ij}]_{m\times m}$ denotes the label kernel matrix based on the predicted label set PL with the kernel function $q(h(\mathbf{x}_i), h(\mathbf{x}_j)) = \langle\psi(h(\mathbf{x}_i)), \psi(h(\mathbf{x}_j))\rangle$. The *ML-HSIC* has the following two advantages: (1) It can effectively evaluate the dependence of TL and PL; (2) The appropriate kernel function can be used to exploit the label correlations. Here, many kernel functions can be applied in P and Q. For example, by using the polynomial kernel of the second degree in the label kernel Q, the second order label correlations can be considered. In this paper, we use RBF kernel in P and Q, since the RBF kernel can potentially exploit the correlations among labels in full order. Therefore, different from conventional accuracy criteria, *ML-HSIC* effectively evaluates the accuracy of multi-label learners with fully considering the correlations among labels.

ML-NCL. Evaluating the diversity of multi-label learners is much more challenging than single-label learning, because, in multi-label learning, the output are a set of labels, instead of a single label. Inspired by the success of Negative Correlation Learning (NCL) in single-label ensemble learning [3,11,12], we propose a criterion to evaluate the diversity of multi-label learners, called *ML-NCL*.

Similar to NCL, the basic idea of *ML-NCL* is to evaluate the negative correlation of each base learner's error with the error for the rest of ensemble. Formally, *ML-NCL* is defined as follows:

$$\begin{aligned} \textit{ML-NCL}(h_j) &= -\sum_{i=1}^{m}\{(h_j(\mathbf{x}_i) - h_{ens}(\mathbf{x}_i))^T \sum_{k\neq j}(h_k(\mathbf{x}_i) - h_{ens}(\mathbf{x}_i))\} \\ &= \sum_{i=1}^{m} \|h_j(\mathbf{x}_i) - h_{ens}(\mathbf{x}_i)\|^2 \end{aligned} \tag{2}$$

where $h_j(\mathbf{x}_i) \in 2^{\mathcal{L}}$ means the output of leaner h_j on data $\mathbf{x}_i$. h_{ens} is the output of the ensemble of N base learners, which is defined as follows:

$$h_{ens}(\mathbf{x}_i) = \frac{1}{N}\sum_{j=1}^{N} h_j(\mathbf{x}_i) \tag{3}$$

The definition shows that *ML-NCL*(h_j) characterizes the significance of difference between the multi-label base learner h_j and the ensemble h_{ens} on training error. Maximizing *ML-NCL* encourages the multi-label base learners to perform differently on training error, so it increases the diversity of base learners. The benefits of *ML-NCL* come from two aspects: (1) It evaluates the diversity of vectors, instead of scale numbers, so *ML-NCL* can be considered as a multi-label version of NCL. (2) It exploits the correlations among multi-label base learners, which has never been done before.

3.2 Multi-objective Optimization Solution

After two criteria are proposed to evaluate the accuracy and diversity of multi-label learners, the next problem is how to optimize them. Different from the single-objective optimization in conventional machine learning, this is a multi-objective optimization problem, i.e. simultaneously maximizing *ML-HSIC* and *ML-NCL*. It can be solved through converting the multi-objective optimization into a single objective optimization by weight sum method. However, this method greatly suffers from the weights setting, because the generalization ability of ensemble largely depends on the trade-off between these two objectives. In this paper, we makes use of an Evolutionary Multi-objective Optimization technology (EMO) [4] to balance the trade-off, since EMO can automatically find optimal trade-off through population evolutionary. Without loss of generality, we focus on solving the multi-objective minimization problem in the following section. The maximization of *ML-HSIC* and *ML-NCL* can be easily converted into a minimization problem.

A good EMO needs to generate a set of solutions that uniformly distributed along the Pareto optimal front [18], which includes two key issues: (1) solutions prone to converge to Pareto optimal front and maintain diversity in the evolutionary process; (2) generating promising solutions in each generation. In order to make EMO fit for multi-label learning, we design many novel mechanisms in the following two sections.

Multi-objective Optimization Mechanism. In this section, we apply the *non-dominated-sort* and *density-assignment* process to make the solutions converge to Pareto optimal front and maintain diversity, respectively.

Non-dominated-sort. The *non-dominated-sort* process sorts solutions according to their raw fitness (i.e. *ML-HSIC* and *ML-NCL*). Instead of the raw fitness, this paper employs the rank-based fitness assignment [7] to reassign the fitness (i.e. a rank value) to the solutions, because the rank-based fitness assignment behaves in a more robust manner. In the rank-based fitness assignment, the solution set is divided into different fronts according to their dominating relations of raw fitness. An example is shown in Figure 2 (*ML-HSIC* and *ML-NCL* are minimized

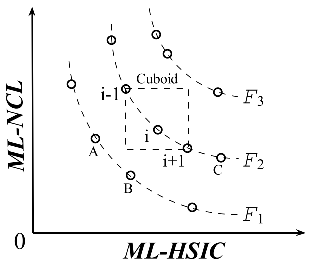


Fig. 2. Illustration of non-dominated-sorting and density-assignment process

here). The solutions in the same front are non-dominated to each other (e.g. solution A and B) and solutions in the higher front are always dominated by some solutions in the lower front (e.g. C in $\mathcal{F}_2$ is dominated by B in $\mathcal{F}_1$). In this way, each solution (i.e. base learner) h_i in a front $\mathcal{F}_a$ has a rank value $h_i^{rank} = a$, and solution h_i is better than solution h_j when $h_i^{rank} < h_j^{rank}$.

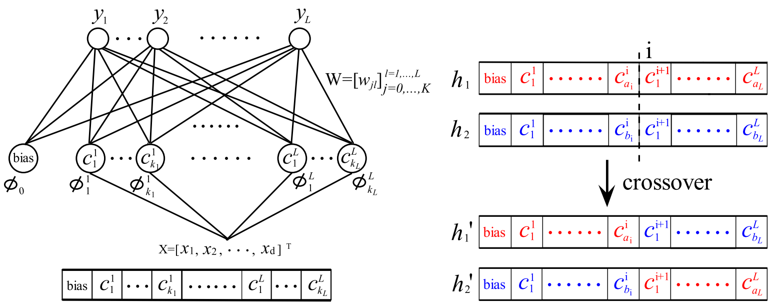
Density-assignment. Along with convergence to the Pareto optimal front, it is also desired that an Evolutionary Algorithm (EA) maintains a good spread of solutions. So the solution in the crowded region is more likely to be deleted. To get a density estimate of solutions surrounding a particular solution in the population, we design the *density-assignment* process that calculates the average distance of two solutions on either side of this solution along each of the objectives. It is simple and effective to estimate the density of solutions. The density estimation of solution h_i, $h_i^{density}$, serves as the perimeter of the cuboid formed by using the nearest neighbors as the vertices. As shown in Figure 2, the density of this i-th solution in its front is the average side length of the cuboid. The small $h_i^{density}$ means solution h_i is in a more crowded region. It implies the solution h_i should be more likely to be deleted.

Select-population. Every solution h_i in the population has two feature values: (1) non-domination rank h_i^{rank}; (2) density estimation $h_i^{density}$, which are determined by the raw fitness *ML-HSIC* and *ML-NCL*. Comprehensively considering both of the features, we define a partial order $\prec$ to compare two solutions. For two solutions h_i and h_j, $h_i \prec h_j$, if and only if

$$h_i^{rank} < h_j^{rank} \ \vee (h_i^{rank} = h_j^{rank} \wedge h_i^{density} > h_j^{density}) \tag{4}$$

That is, between two solutions with different non-domination ranks, we prefer the solution with the lower rank. Otherwise, if both solutions belong to the same front, then we prefer the solution that is located in a less crowded region. After sorting the population with $\prec$, *select-population* process selects top solutions, and guarantees that good solutions will be kept.

Base Learner and Evolutionary Operators. In the framework of EnML, many multi-label base learners can be used, such as HMC tree [19], BP-MLL


y_1
y_2
y_L
W=$[w_{jl}]_{j=0,\ldots,K}^{l=1,\ldots,L}$
bias
c_1^1
$c_{k_1}^1$
c_1^L
$c_{k_L}^L$
ϕ_0
ϕ_1^1
$\phi_{k_1}^1$
ϕ_1^L
$\phi_{k_L}^L$
X=$[x_1, x_2, \cdots, x_d]^T$
bias | c_1^1 | ⋯ | $c_{k_1}^1$ | ⋯⋯ | c_1^L | ⋯ | $c_{k_L}^L$
i
h_1 bias | c_1^1 | ⋯⋯ | $c_{a_i}^i$ | c_1^{i+1} | ⋯⋯ | $c_{a_L}^L$
h_2 bias | c_1^1 | ⋯⋯ | $c_{b_i}^i$ | c_1^{i+1} | ⋯⋯ | $c_{b_L}^L$
crossover
h_1' bias | c_1^1 | ⋯⋯ | $c_{a_i}^i$ | c_1^{i+1} | ⋯⋯ | $c_{b_L}^L$
h_2' bias | c_1^1 | ⋯⋯ | $c_{b_i}^i$ | c_1^{i+1} | ⋯⋯ | $c_{a_L}^L$

(a) Genetic representation (b) Crossover operation

Fig. 3. (a) Architecture of RBF and its corresponding genetic representation. (b) The crossover operation. The crossover point i is selected between two prototype vectors.

[23] and ML-RBF [21]. Different types of base learners will lead to different genetic representation and operation. Because the structure can be effectively encoded and the weights can be efficiently calculated in close form, we select the RBF neural network in ML-RBF [21] as the multi-label base learner in EnML, however an additional regularization term is added to reduce overfitting risks.

The architecture of RBF is shown in Figure 3(a). It is described as follows: (1) The input of a RBF corresponds to a d-dimension feature vector. (2) The hidden layer of RBF is composed of L sets of prototype vectors, i.e. $\bigcup_{l=1}^{L} C_l$. Here, C_l consists of k_l prototype vectors $\{c_1^l, c_2^l, \cdots, c_{k_l}^l\}$. For each class $l \in \mathcal{L}$, the k-means clustering is performed on the set of instances U_l with label l. Thereafter, k_l clustered groups are formed for class l and the j-th centroid $(1 \leq j \leq k_l)$ is regarded as a prototype vector c_j^l of basis function $\phi_j^l(\cdot)$. (3) Each output neuron is related to a possible class. In the hidden layer of RBF, the number of clusters k_l is a fraction α of the number of instances in U_l:

$$k_l = \alpha \times |U_l| \tag{5}$$

The scale coefficient α controls the structure and complexity of RBF base learner.

Different from the error function in original RBF, we add a regularization term into the error function. The regularization term greatly reduces the overfitting risk and improves the stability of solutions as observed in the experiments.

$$E = \frac{1}{2}\sum_{i=1}^{m}\sum_{l=1}^{L}(y_l(\mathbf{x}_i) - t_l^i)^2 + \gamma\sum_{j=0}^{K}\sum_{l=1}^{L} w_{jl}^2 \tag{6}$$

where $y_l(\mathbf{x}_i)$ represents the predicted value of instance $\mathbf{x}_i$ on label l, t_l^i is the real value of instance i on label l, $K = \sum_{l=1}^{L} k_l$, and γ is the regularization coefficient. Similar to the derivation of minimizing the error function by scaled-conjugate-gradient descent in [3], the optimal output weights W can be computed in closed form by

$$W = (\Phi'\Phi + \gamma I)^{-1}\Phi' T \tag{7}$$

Here $\Phi = [\phi_{ij}]_{m\times(K+1)}$ with elements $\phi_{ij} = \phi_j(\mathbf{x}_i)$, $W = [w_{jl}]_{(K+1)\times L}$ with elements w_{jl}, and $T = [t_{il}]_{m\times L}$ with elements $t_{il} = t_l^i$. Through extensive experiments, the regularization coefficient γ is fixed at 0.1 in this paper.

Genetic representation. According to the structure of RBF, we propose a novel genetic representation that is the sequence of prototypes $\{bias, c_1^1, c_1^2, \cdots c_{k_L}^L\}$. An example is shown in Figure 3(a). The genetic representation has the following advantages. (1) When the prototypes (c) are determined, the basis functions (ϕ) and the weights (W) can be efficiently computed. It means the performance of RBF mostly depends on the selection of the prototypes. (2) It is easy to design the crossover and mutation operators by tuning these prototypes.

Initialization. When the base learner is RBF, the initialization operation of EnML generates a set of RBF learners with different scale coefficient α (see Equation 5). As suggested in [21], α is randomly selected from [0.01, 0.02] in the experiments, which generates a set of RBF base learners with different structures.

Generate-offspring. Generating new solutions is realized by the *generate-offspring* process. The basic idea is to randomly select parent solutions from the current population based on the roulette wheel selection [1,3] and do crossover and mutation operation to generate new solutions with the ratio of *cro_Rat* and $1-$*cro_Rat* respectively. Following the general rule in EA, *cro_Rat* is fixed at 0.8 in this paper.

The roulette wheel selection [1,3] assigns each solution with the appropriate selection pressure, and guarantees the better solution with a high and appropriate selected probability. This paper adapts the cut and splice crossover [9] which randomly chooses a crossover point for two RBFs and swaps their prototypes beyond this point. Different from traditional cut and splice crossover, the crossover point in EnML is randomly selected between two prototype vectors, rather than in a arbitrary position. It guarantees that each prototype vector in the newly generated RBF is unabridged cluster centroids. Figure 3(b) shows an example of crossover operation. The mutation operator randomly selects some prototype vectors in a RBF, and does the following two structural mutation operations with the same probability. (1) Randomly delete a prototype. (2) Add one prototype whose center is determined by a random combination of all centroids in this prototype vector. The width of the centroid of the new RBF is recalculated as in [21]. The weights are calculated following Equation 7.

3.3 Algorithm Framework

EnML is described in Algorithm 1. In the model training phase, EnML transforms the optimized objectives (i.e. *ML-HSIC* and *ML-NCL*) to a fitness measure by the creation of a number of fronts, sorted according to *non-dominated-sort*. After the fronts have been created, *density-assignment* assigns its members with a density value later to be used for diversity maintenance. In each generation, N new solutions are generated with *generate-offspring*. Of the $2N$ solutions, *select-population* selects the N best solutions for the next generation. In this way, a huge elite can be kept and optimized from generation to generation. In the testing phase, all solutions predict labels of unseen data and combine their

Algorithm 1. EnML

```
Input:
$\mathcal{D}$: training data    $\mathcal{U}$: testing data    $h$: base learner
$N$: # of base learners    $G$: # of generations
output:
$Y(\mathbf{x})$: predicted labels for instance $\mathbf{x} \in \mathcal{U}$
procedure Training
    generate $P_0 = \{h_1, h_2, \cdots, h_N\}$ on $\mathcal{D}$ at random
    $P_1 = (\mathcal{F}_1, \mathcal{F}_2, \cdots) = \textit{non-dominated-sort}(P_0)$
    for $t = 1 : G$ do
        $Q_t = \textit{generate-offspring}(P_t)$
        $R_t = P_t \bigcup Q_t$
        $F = (\mathcal{F}_1, \mathcal{F}_2, \cdots) = \textit{non-dominated-sort}(R_t)$
        $\textit{density-assignment}(F)$
        $P_{t+1} = \textit{select-population}(F)$
        $t = t + 1$
    end for
end procedure
procedure Testing
    For $\mathbf{x} \in \mathcal{U}$, label set $Y(\mathbf{x}) = \{l | \frac{1}{N} \sum_{i=1}^{N} h_i(\mathbf{x}, l) > 0; h_i \in P_t, l \in \mathcal{L}\}$
end procedure
```

Table 1. Summary of experimental datasets

	Dataset						
Characteristic	Image	Yeast	Arts	Health	Science	Recreation	Entertain.
# of instances	2000	2417	5000	5000	5000	5000	5000
# of features	294	103	462	612	743	606	640
# of labels	5	14	26	32	40	22	21
domain	biology	media	text	text	text	text	text

results with a simple vote. Note that EnML can not only optimize *ML-HSIC* and *ML-NCL* but also directly optimize either of these two objectives.

4 Experiments

4.1 Experimental Setup

Data Collections: We tested our algorithm on seven real-world multi-label classification datasets from three different domains as summarized in Table 1. The first dataset is *Yeast* [15,21,23,25] in biology, where the task is to predict the gene functional classes of the Yeast Saccharomyces cerevisiae. The second dataset *Image* [15,21,23,25] involves the task of automatic image annotation for scene images. The other five dataset are from Yahoo [21,24], where the task is to predict topic categories of each text document.

Evaluation Metrics: Here we adopt five state-of-the-art multi-label evaluation metrics which are most popular in the literature. Assume we have a multi-label dataset $\mathcal{U}$ containing n multi-label instances $(\mathbf{x}_i, Y_i)$. Let $h(\mathbf{x}_i)$ denote the predicted label set of a multi-label learner h for $\mathbf{x}_i$, and the real-valued function

$f(\mathbf{x}_i, y_l) \in R$ represents the ranking quality score of learner h on label y_l for input $\mathbf{x}_i$. We have the following evaluation criteria:

• *hamming loss* [5,22]: evaluates the number of labels whose relevance is incorrectly predicted.

$$hammingloss(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{L}\|h(\mathbf{x}_i) \oplus Y_i\|_1 \tag{8}$$

where $\bigoplus$ stands for the symmetric difference of two sets (XOR operation), and $\|.\|_1$ denotes the l_1-norm. The smaller the value, the better the performance.

• *ranking loss* [6,25]: evaluates the average fraction of label pairs that are misordered for the instance.

$$rankingloss(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i||\overline{Y_i}|}|R_i| \tag{9}$$

where $R_i = \{(y_1, y_2)|f(\mathbf{x}_i, y_1) \leq f(\mathbf{x}_i, y_2), (y_1, y_2) \in Y_i \times \overline{Y_i}\}$. Here $\overline{Y_i}$ denotes the complementary set of Y_i in Y. The smaller the value, the better the performance.

• *one-error* [6,25]: evaluates how many times the top-ranked label is not in the set of proper labels of the instance.

$$one\text{-}error(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n}[\![[argmax_{y\in\mathcal{L}} f(\mathbf{x}_i, y)] \notin Y_i]\!] \tag{10}$$

Here for predicate π, $[\![\pi]\!]$ equals 1 if π holds and 0 otherwise. The smaller the value, the better the performance.

• *coverage* [6,25]: evaluates how many steps are needed, on average, to move down the ranked label list in order to cover all the proper labels of the instance.

$$coverage(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n} max_{y\in Y_i} rank^f(\mathbf{x}_i, y) - 1 \tag{11}$$

$rank^f(\cdot,\cdot)$ is derived from the real-valued function $f(\cdot,\cdot)$. If $f(\mathbf{x}_i, y_1) > f(\mathbf{x}_i, y_2)$, then $rank^f(\mathbf{x}_i, y_1) < rank^f(\mathbf{x}_i, y_2)$. The smaller the value, the better the performance.

• *average precision* [6,25]: evaluates the average fraction of proper labels ranked above a particular label $y \in Y_i$.

$$avgprec(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{|P_i|}{rank^f(\mathbf{x}_i, y)} \tag{12}$$

where $P_i = \{y'|rank^f(\mathbf{x}_i, y') \leq rank^f(\mathbf{x}_i, y), y' \in Y_i\}$. The larger the value, the better the performance.

Compared Methods: In order to test performance of our proposed EnML, we do comprehensive comparison with the most representative multi-label learning approaches, including ML-RBF [21], the base learner of our approach, and two ensemble based approaches: ECC [15] and RAKEL [17]. In addition, in order to validate the effectiveness of two objective functions, we include two special cases of EnML that only optimize one single objective (i.e. *ML-HISC* or *ML-NCL*). These approaches are briefly summarized as follows.

- EnML: the proposed approach in this paper. It simultaneously optimizes two objectives: *ML-HSIC* and *ML-NCL*.
- EnML$_{HSIC}$: a special case of EnML, which only optimizes *ML-HSIC*.
- EnML$_{NCL}$: a special case of EnML, which only optimizes *ML-NCL*.
- ML-RBF [21]: the multi-label learning algorithm based on RBF neural network, which is also the base learner we use in EnML.
- ECC [15]: an ensemble method for multi-label learning based on the bagging of classifier chains.
- RAKEL [17]: another ensemble method for multi-label learning, where the single-label base learner is trained for a small random subset of labels.

In order to fit for EnML as a minimization problem, we convert the original objectives into an equivalent minimization problem as follows:

$$\begin{aligned} \textit{ML-HSIC}' &= 1/log(\textit{ML-HSIC}) \\ \textit{ML-NCL}' &= 1 - \textit{ML-NCL}/(m \times L) \end{aligned} \tag{13}$$

Note that the two new objectives both need to be minimized and fall in [0,1], such that it is convenient to perform *non-dominated-sort* and *density-estimate* process in our evolutionary multi-objective optimization algorithm. As in [21], ML-RBF is implemented with fixed default parameters ($\alpha = 0.01$ and $\mu = 1.0$). For ECC, the ensemble size is set to 10 and sampling ratio is set to 67% as suggested in the literature [15]. For RAKEL [17], we always set the parameter k as $\frac{|L|}{2}$ to provide the highest accuracy. The population size and running generation of EnML based approaches are set as 30 and 10 respectively in all experiments.

4.2 Performance Comparison

We perform ten-fold cross-validation on each experimental dataset. On each dataset, we report the mean values performance and standard deviations for each algorithm with the rank based on its results indicated in the parentheses. All experiments are conducted on machines with Intel Xeon Quad-Core CPUs of 2.26 GHz and 24 GB RAM.

The performances of six algorithms are shown in Table 2 to Table 6. It is clearly shown that EnML significantly outperforms the other baseline methods, including the non-ensemble method ML-RBF and two ensemble methods ECC and RAKEL, on all criteria and datasets. The small standard deviations of the rank values of EnML (ranging from 0 to 0.49) also indicate the superior of EnML is consistent on all datasets and evaluated criteria. The results illustrate that the

Table 2. Performance (mean±std.(rank)) of each algorithm in terms of *hamming loss*. Ave. Rank represents the mean and standard deviation of the rank values of each algorithm in all datasets.

	Algorithm					
Dataset	EnML	ML-RBF	ECC	RAKEL	EnML$_{HSIC}$	EnML$_{NCL}$
Image	0.1603±0.0058(2)	0.1653±0.0067(3)	0.1786±0.0108(6)	0.1724±0.0117(5)	0.1586±0.0065(1)	0.1665±0.0051(4)
Yeast	0.1889±0.0052(2)	0.1935±0.0058(4)	0.2056±0.0082(5)	0.2287±0.0105(6)	0.1887±0.0064(1)	0.1894±0.0059(3)
Arts	0.0531±0.0014(2)	0.0542±0.0016(4)	0.0754±0.0045(6)	0.0612±0.0013(5)	0.0528±0.0014(1)	0.0538±0.0015(3)
Health	0.0316±0.0016(2)	0.0331±0.0016(4)	0.0361±0.0021(5)	0.0373±0.0016(6)	0.0314±0.0017(1)	0.0322±0.0017(3)
Science	0.0317±0.0008(2)	0.0324±0.0009(4)	0.0424±0.0054(6)	0.0360±0.0016(5)	0.0313±0.0010(1)	0.0320±0.0008(3)
Recreation	0.0543±0.0023(2)	0.0555±0.0022(4)	0.0688±0.0055(6)	0.0589±0.0028(5)	0.0539±0.0021(1)	0.0553±0.0025(3)
Entertain.	0.0502±0.0016(2)	0.0512±0.0016(3)	0.0654±0.0053(6)	0.0587±0.0030(5)	0.0496±0.0013(1)	0.0514±0.0012(4)
Ave. Rank	2.00±0.00	3.71±0.49	5.71±0.49	5.29±0.49	1.00±0.00	3.29±0.49

Table 3. Performance (mean±std.(rank)) of each algorithm in terms of *ranking loss*

	Algorithm					
Dataset	EnML	ML-RBF	ECC	RAKEL	EnML$_{HSIC}$	EnML$_{NCL}$
Image	0.1478±0.0112(1)	0.1558±0.0121(4)	0.2411±0.0153(6)	0.1765±0.0200(5)	0.1485±0.0112(2)	0.1536±0.0106(3)
Yeast	0.1597±0.0083(1)	0.1621±0.0073(4)	0.2776±0.0223(6)	0.2179±0.0156(5)	0.1603±0.0087(2)	0.1619±0.0073(3)
Arts	0.1119±0.0099(1)	0.1131±0.0098(3)	0.3814±0.0251(6)	0.2589±0.0106(5)	0.1150±0.0104(4)	0.1124±0.0093(2)
Health	0.0482±0.0057(1)	0.0496±0.0051(3)	0.2401±0.0130(6)	0.1822±0.0125(5)	0.0505±0.0056(4)	0.0490±0.0054(2)
Science	0.0957±0.0072(1)	0.1002±0.0071(3)	0.3840±0.0238(6)	0.2854±0.0138(5)	0.1017±0.0079(4)	0.0992±0.0072(2)
Recreation	0.1216±0.0101(1)	0.1253±0.0099(3)	0.3434±0.0203(6)	0.2874±0.0227(5)	0.1257±0.0118(4)	0.1229±0.0095(2)
Entertain.	0.0913±0.0070(1)	0.0946±0.0073(3)	0.2926±0.0193(6)	0.2874±0.0221(5)	0.0949±0.0073(4)	0.0933±0.0062(2)
Ave. Rank	1.00±0.00	3.29±0.49	6.00±0.00	5.00±0.00	3.43±0.98	2.29±0.49

Table 4. Performance (mean±std.(rank)) of each algorithm in terms of *one error*

	Algorithm					
Dataset	EnML	ML-RBF	ECC	RAKEL	EnML$_{HSIC}$	EnML$_{NCL}$
Image	0.2735±0.0236(2)	0.2860±0.0299(4)	0.2935±0.0249(5)	0.3065±0.0335(6)	0.2695±0.0247(1)	0.2815±0.0208(3)
Yeast	0.2156±0.0235(1)	0.2189±0.0175(3)	0.2742±0.0218(5)	0.2751±0.0300(6)	0.2193±0.0286(4)	0.2160±0.0210(2)
Arts	0.4400±0.0134(2)	0.4512±0.0124(4)	0.4734±0.0291(5)	0.5470±0.0137(6)	0.4314±0.0177(1)	0.4450±0.0130(3)
Health	0.2416±0.0204(2)	0.2482±0.0250(4)	0.2430±0.0183(3)	0.2946±0.0184(6)	0.2398±0.0230(1)	0.2494±0.0219(5)
Science	0.4862±0.0185(2)	0.5016±0.0181(5)	0.5008±0.0432(4)	0.5784±0.0199(6)	0.4794±0.0174(1)	0.4916±0.0187(3)
Recreation	0.4492±0.0159(2)	0.4542±0.0220(3)	0.4618±0.0196(5)	0.5304±0.0281(6)	0.4398±0.0187(1)	0.4548±0.0132(4)
Entertain.	0.3824±0.0241(2)	0.3916±0.0251(5)	0.3836±0.0309(3)	0.4746±0.0278(6)	0.3816±0.0222(1)	0.3914±0.0234(4)
Ave. Rank	1.86±0.38	4.00±0.82	4.29±0.95	6.00±0.00	1.47±1.13	3.43±0.98

Table 5. Performance (mean±std.(rank)) of each algorithm in terms of *coverage*

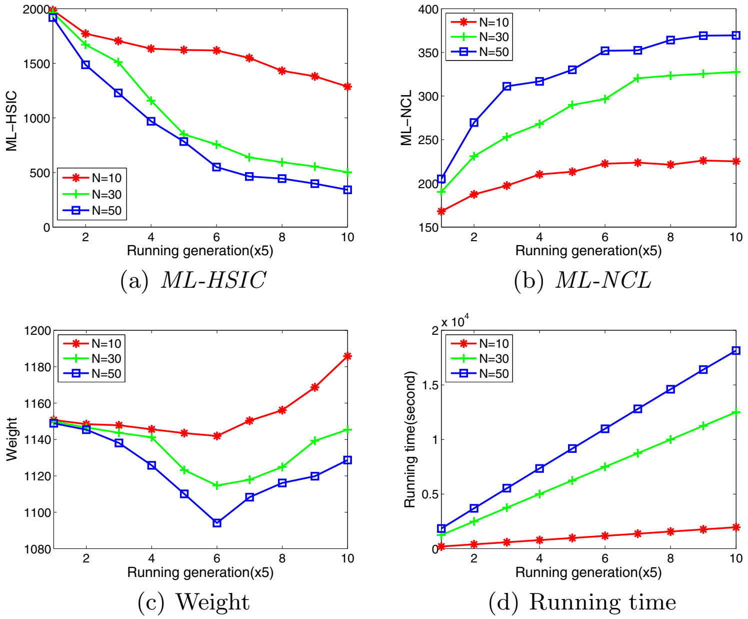
	Algorithm					
Dataset	EnML	ML-RBF	ECC	RAKEL	EnML$_{HSIC}$	EnML$_{NCL}$
Image	0.8740±0.0548(2)	0.8955±0.0562(4)	0.9715±0.0776(5)	0.9795±0.0831(6)	0.8570±0.0487(1)	0.8900±0.0468(3)
Yeast	6.1845±0.1465(1)	6.2465±0.1433(4)	7.1431±0.2688(5)	7.5347±0.2521(6)	6.2138±0.1353(2)	6.2453±0.1384(3)
Arts	4.5738±0.4115(1)	4.6116±0.3783(3)	7.8582±0.4686(5)	8.8862±0.3652(6)	4.7192±0.4110(4)	4.5808±0.3720(2)
Health	3.1998±0.3144(1)	3.2280±0.2797(3)	8.2418±0.3797(5)	8.7686±0.4684(6)	3.2930±0.2942(4)	3.2122±0.3042(2)
Science	5.5188±0.4325(1)	5.6016±0.4341(3)	11.403±0.4453(5)	13.744±0.6340(6)	5.7114±0.4432(4)	5.5476±0.4108(2)
Recreation	3.6858±0.2916(1)	3.7452±0.2983(3)	6.2390±0.4696(5)	7.6552±0.6209(6)	3.7790±0.3304(4)	3.6872±0.2831(2)
Entertain.	2.7686±0.1832(1)	2.8102±0.1739(3)	5.7008±0.2569(5)	7.1750±0.4761(6)	2.8478±0.1885(4)	2.7796±0.1434(2)
Ave. Rank	1.14±0.38	3.29±0.49	5.00±0.00	6.00±0.00	3.29±1.25	2.29±0.49

Table 6. Performance (mean±std.(rank)) of each algorithm in terms of *average precision*

	Algorithm					
Dataset	EnML	ML-RBF	ECC	RAKEL	EnML$_{HSIC}$	EnML$_{NCL}$
Image	0.8288±0.0144(1)	0.8118±0.0145(4)	0.7977±0.0148(5)	0.7952±0.0215(6)	0.8226±0.0141(2)	0.8139±0.0122(3)
Yeast	0.7754±0.0146(1)	0.7720±0.0133(4)	0.7313±0.0236(5)	0.7170±0.0165(6)	0.7747±0.0152(2)	0.7734±0.0127(3)
Arts	0.6433±0.0113(2)	0.6366±0.0116(4)	0.5613±0.0149(5)	0.5122±0.0138(6)	0.6473±0.0123(1)	0.6406±0.0112(3)
Health	0.7988±0.0135(2)	0.7941±0.0151(4)	0.7247±0.0115(5)	0.6986±0.0142(6)	0.7994±0.0142(1)	0.7957±0.0132(3)
Science	0.6128±0.0152(2)	0.6026±0.0155(4)	0.5328±0.0227(5)	0.4712±0.0213(6)	0.6178±0.0172(1)	0.6090±0.0167(3)
Recreation	0.6501±0.0159(2)	0.6435±0.0170(4)	0.5770±0.0145(5)	0.5355±0.0242(6)	0.6520±0.0164(1)	0.6448±0.0134(3)
Entertain.	0.7028±0.0146(2)	0.6971±0.0169(4)	0.6338±0.0151(5)	0.5763±0.0232(6)	0.7029±0.0142(1)	0.6976±0.0143(3)
Ave. Rank	1.71±0.49	4.00±0.00	5.00±0.00	6.00±0.00	1.29±0.49	3.00±0.00

ensemble in our EnML can effectively improve the generalization performance in multi-label learning, compared to non-ensemble methods (e.g. ML-RBF). In addition, the superior of EnML over those ensemble methods for multi-label learning (e.g. ECC and RAKEL) also confirms our assumption: the ensemble of multi-label base learners is more effective to improve the generalization ability of multi-label learning system than the ensemble of single-label base learners. We think one of the important reasons behind the performance improvement of EnML lies in our EnML emphasizes the diversity of multi-label base learners by explicitly optimizing a diversity-related objective, which has never been done in multi-label learning so far.

Then we further study the effect of objective functions in our EnML method on the performances by comparing EnML with EnML$_{HSIC}$ and EnML$_{NCL}$. From Table 2 to Table 6, we can also observe that the three versions of EnML rank top three on most criteria and they always have the best performance on each dataset. By optimizing the diversity-related objective *ML-NCL*, EnML$_{NCL}$ generates a set of diverse base learners, so EnML$_{NCL}$ outperforms the base learner ML-RBF on most criteria. However, without optimizing the accuracy of individual base learner, EnML$_{NCL}$ performs worse than EnML on all criteria. Although EnML$_{HSIC}$ can achieve a little better performances than EnML in *hamming loss*, *one-error*, and *average precision* on some datasets, however, on the other two criteria, *ranking loss* and *coverage*, EnML$_{HSIC}$ is not only worse than EnML and EnML$_{NCL}$, but also worse than the base learner ML-RBF. It can be explained that EnML$_{HSIC}$ optimizes the accuracy-related objective *ML-HSIC*, which makes it perform well on the *ML-HSIC* related criteria, such as *hamming loss*, *one-error*, and *average precision.* However, without emphasizing the diversity of base learners, the optimal base learners obtained by EnML$_{HSIC}$ can be very similar with each other, thus the generalization ability of the ensemble can be weak. By considering the accuracy and diversity objectives simultaneously, EnML can obtain a group of accurate and diverse multi-label base learners and the population evolutionary strategy in EnML automatically finds the optimal trade-off between these two objectives. As a consequence, EnML consistently improves the generalization ability of multi-label ensemble, thus it comprehensively boosts the multi-label classification performances.



(a) *ML-HSIC*

(b) *ML-NCL*

(c) Weight

(d) Running time

Fig. 4. The evolutionary performances of EnML with different parameter settings

4.3 Parameter Settings

In this section, we study the effects of the parameters in our EnML method. There are two genetic operation related parameters in EnML, i.e. the population size N and the running generation G. We perform the experiment on 2000 instances of the Arts dataset in Yahoo dataset collection [21,24] with ten-fold cross-validation under different parameter configurations. Specifically, when the population size N is set as 10, 30, and 50, we report the average of objective values, running time and weights (sum of absolute values in W). The results are shown in Figure 4.

From Figure 4(a) and (b), we can clearly observe that *ML-NCL* goes up but *ML-HSIC* goes down when the running generation increases. The different trend of these two objectives indicate that they have the intrinsic conflict. It is not surprising. The maximization of the *ML-HSIC* guides the predicted labels of base learners to converge to the real labels. So it makes these base learners identical. However, the maximization of the *ML-NCL* encourages base learners to be as diverse as possible on the training error. Therefore, these two objectives are naturally conflicting. The conflict makes EnML seek to find a good balance between the two objectives by population optimization. Note that here *ML-HSIC* and *ML-NCL* just evaluate the accuracy and diversity of base learners, not the performance of the ensemble. The decrease of *ML-HSIC* does not mean the degradation of the ensemble. In fact, the increase of *ML-NCL* shows that base learners become more diverse, which helps to improve the performance of the ensemble. Figure 4(c) shows that the weight goes down and then goes up when the running generation increases. We think the reason is that *ML-NCL* helps to control the model complexity. However, when the running generation becomes too large, these learners become more complex, and thus their weights increase. If we do not add the regularization term in the error function of RBF (see Equation 6), the weights will increase sharply, which means these learners are overfitting. Figure 4(d) illustrates that the running time of EnML increases linearly with the population size N and running generation G.

5 Conclusion

In this paper, we first study the multi-label ensemble learning problem which aims at building a set of accurate and diverse multi-label base learners to improves the generalization ability of multi-label learning system. In order to solve this problem, we propose a novel solution EnML. With an evolutionary multi-objective optimization method, EnML simultaneously optimizes two objective functions that evaluate the accuracy and diversity of multi-label learners, respectively, and constructs a set of accurate and diverse multi-label base learners to make predictions. Extensive experiments show that EnML can effectively improve the generalization ability of multi-label learning system and thus boosts the predictive performance for multi-label classification.

Acknowledgments. It is supported by the National Natural Science Foundation of China (No. 60905025, 61074128, 61035003) and the 863 Program of China No.2009AA04Z136.

References

1. Baker, J.: Adaptive Selection Methods for Genetic Algorithms. In ICGA, pp. 100–111 (1985)
2. Breiman, L.: Bagging Predictors. Machine Learning 24(2), 123–140 (1996)
3. Chen, H., Yao, X.: Multiobjective Neural Network Ensembles Based on Regularized Negative Correlation Learning. Transactions on Knowledge and Data Engineering 22(12), 1738–1751 (2010)
4. Deb, K.: Multiobjective Optimization using Evolutionary Algorithms. Wiley, UK (2001)
5. Dembczynski, K., Cheng, W., Hullermeier, E.: Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In: ICML, pp. 279–286 (2010)
6. Elisseeff, A., Weston, J.: A Kernel Method for Multilabelled Classification. In: NIPS, pp. 681–687 (2002)
7. Goldberg, D.E.: Generic Algorithms in Search Optimization and Machine Learning, USA, Boston (1989)
8. Gretton, A., Bousquet, O., Smola, A., Scholkopf, B.: Measuring Statistical Dependence with Hilbert-Schmidt Norms. In: Jain, S., Simon, H.U., Tomita, E. (eds.) ALT 2005. LNCS (LNAI), vol. 3734, pp. 63–77. Springer, Heidelberg (2005)

9. Goldberg, D., Deb, K., Kargupta, H., Harik, G.: Rapid, Accurate Optimization of Difficult Problems using Fast Messy Genetic Algorithms. In: ICGA, pp. 56–64 (1993)
10. Krogh, A., Vedelsby, J.: Neural Network Ensembles, Cross Validation, and Active Learning. In: NIPS, pp. 231–238 (1995)
11. Liu, Y., Yao, X.: Ensemble Learning via Negative Correlation. Neural Networks 12(10), 1399–1404 (1999)
12. Liu, Y., Yao, X.: Simultaneous Training of Negatively Correlated Neural Networks in an Ensemble. Transaction on Systems, Man, and Cybernetics, Part B: Cybernetics 29(6), 716–725 (1999)
13. Petterson, J., Caetano, T.: Reverse Multi-label Learning. In: NIPS (2010)
14. Read, J., Pfahringer, B., Holmes, G.: Multi-label Classification using Ensembles of Pruned Sets. In: ICDM, pp. 995–1000 (2008)
15. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier Chains for Multi-label Classification. In: ECML, pp. 254-269 (2009)
16. Tsoumakas, G., Katakis, I., Vlahavas, I. P.: Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In: ECML/PKDD Workshop (2008)
17. Tsoumakas, G., Vlahavas, I.P.: Random k-Labelsets: an Ensemble Method for Multilabel Classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 406–417. Springer, Heidelberg (2007)
18. Veldhuizen, D.A.V., Lamont, G.B.: Multiobjective Evolutionary Algorithms: Analyzing the state-of-the-art. Evolutionary Computation 18(2), 125–147 (2000)
19. Vens, C., Struyf, J., Schietgat, L., Dzeroski, S., Blockeel, H.: Decision Tree for Hierarchical Multi-label Classification. Machine Learning 2(73), 185–214 (2008)
20. Yang, B.S., Sun, J.T., Wang, T.J., Chen Z.: Effective Multi-label Active Learning for Text Classification. In: KDD, pp. 917–925 (2009)
21. Zhang, M.L.: ML-RBF: RBF Neural Networks for Multi-label Learning. Neural Process Letters 29(2), 61–74 (2009)
22. Zhang, X., Yuan, Q., Zhao, S., Fan, W., Zheng, W., Wang, Z.: Multi-label Classification without the Multilabel Cost. In: SDM, pp. 778–789 (2010)
23. Zhang, M.L., Zhou, Z.H.: Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. Transactions on Knowledge and Data Engineering 18(10), 1338–1351 (2006)
24. Zhang, M.L., Zhou, Z.H.: Ml-knn: a Lazy Learning Approach to Multi-label Learning. Pattern Recognition 40(7), 2038–2048 (2007)
25. Zhang, M.L., Zhang, K.: Multi-label Learning by Exploiting Label Dependency. In: KDD, pp. 999–1007 (2010)

Rule-Based Active Sampling for Learning to Rank

Rodrigo Silva, Marcos A. Gonçalves, and Adriano Veloso

Department of Computer Science, Federal University of Minas Gerais
{rmsilva,mgoncalv,adrianov}@dcc.ufmg.br

Abstract. Learning to rank (L2R) algorithms rely on a labeled training set to generate a ranking model that can be later used to rank new query results. Producing these labeled training sets is usually very costly as it requires human annotators to assess the relevance or order the elements in the training set. Recently, active learning alternatives have been proposed to reduce the labeling effort by selectively sampling an unlabeled set. In this paper we propose a novel rule-based active sampling method for Learning to Rank. Our method actively samples an unlabeled set, selecting new documents to be labeled based on how many relevance inference rules they generate given the previously selected and labeled examples. The smaller the number of generated rules, the more dissimilar and more "informative" is a document with regard to the current state of the labeled set. Differently from previous solutions, our algorithm does not rely on an initial training seed and can be directly applied to an unlabeled dataset. Also in contrast to previous work, we have a clear stop criterion and do not need to empirically discover the best configuration by running a number of iterations on the validation or test sets. These characteristics make our algorithm highly practical. We demonstrate the effectiveness of our active sampling method on several benchmarking datasets, showing that a significant reduction in training size is possible. Our method selects as little as 1.1% and at most 2.2% of the original training sets, while providing competitive results when compared to state-of-the-art supervised L2R algorithms that use the complete training sets.

1 Introduction

Many applications need ranking functions to order results before presenting them to their users, mainly when there is a large potential number of candidate results. Search engines and product recommendation systems, for instance, need to rank results based on their estimated relevance with respect to a query or based on a user profile and/or personal preferences. In the last few years, there has been considerable interest in the research community for machine learning techniques to learn to rank lists of documents or other data effectively [20]. Learning to Rank algorithms, which deliver superior performance when compared to more traditional approaches such as BM25 [13], use labeled training sets to build ranking models that are used to rank results at query time. The effectiveness of the learned functions is usually directly correlated with the amount of supervised training data available [9]. Yet it is very costly and laborious to produce training sets containing labeled instances, since they must be assessed by a human annotator. Some of the benchmarking datasets we use in our experimental evaluation, for

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 240–255, 2011.

example, have training sets comprised of almost 90,000 labeled instances. In most real-life settings it is unfeasible to create such large sets from scratch.

Active learning techniques have been given much attention lately to help deal with the labeling effort problem [2,3,9,21]. These techniques are used to actively sample an unlabeled dataset to select instances that maximize the effectiveness of learned rank functions. By carefully selecting and labeling instances it may be possible to use a (very) small training set and yet achieve highly effective learned functions, thus minimizing the labeling effort. The rationale behind active learning is that the instances selected are representative of the document corpus and somehow more "informative" to the learning algorithm. By using only a highly informative set, there may be a reduction of noise and uncertainty in the learning process, possibly yielding even more effective functions than those obtained using a large training set. In most active learning methods, samples are selected incrementally from an unlabeled set and are used to update an already existing learning function. At each step the most "informative" instances are selected to be labeled and added to the training set. Several strategies and heuristics have been proposed in the literature on how to determine the most informative samples. In some methods, the most ambiguous - or those instances for which the learner is most *uncertain* about - are selected [7]. In other settings, a query-by-committee (QBC) strategy is employed where competing learners vote on the label of the candidate samples and those about which they most disagree are selected [17]. Some algorithms select the samples that would cause the greatest change to the currently learned function [16]. Other methods select instances that would lead to the minimal expected future error or, similarly, optimize some other performance measure such as precision or recall [15].

There has been work proposing active learning methods for classification tasks [4,10,11]. While classification functions output an specific class for each data item, ranking functions must produce partial orders of items either through some scoring function, pairwise ordering or listwise ordering of the items. Most active sampling methods for classification try to directly minimize the classification error, but this approach is not straightforward to extend to the ranking problem since position-based measures such as MAP and NDCG are usually non-continuous and non-differentiable [8]. Additionally, in most supervised learning settings, samples can be treated as independent of each other which is not the case for L2R where each sample represents a document *relative to a query*. Thus, in L2R, instances are conditionally independent given a query [9].

In this paper we propose a new active sampling technique based on association rules. Learning to rank using demand-driven association rules has been shown to provide competitive ranking quality [19]. The method proposed here uses association rules to actively select documents[1] from an unlabeled set based on how many inference rules are generated for each document. At each step, a new document is chosen for labeling as the most "dissimilar" with respect to the already selected samples (but with no need to create a model), with the goal of increasing diversity and representativeness. This is performed by choosing from the unlabeled data the document u_i that generates the smallest number of rules when considering a "projection" of the current selected training set with respect to the features of u_i. This projection aims at removing examples from the current training set that are not useful in producing rules for u_i.

[1] We use the terms documents, instances, samples, and examples interchangeably in this paper.

The more dissimilar the candidate document, the fewer the rules generated for it, meaning that few already labeled instances in the projected training set share common feature-values with the candidate. This process is repeated until the algorithm converges, which happens when an already selected document is selected again, i.e., there is no more information useful for generating rules from any other document in the unlabeled set.

Despite being very effective in several cases, as demonstrated in our experiments, our method may sometimes converge too fast, resulting in a very small labeled training set. We propose a strategy to delay this convergence and increase the size and diversity of the labeled set which consists in vertically partitioning the features in the unlabeled set, generating in practice several reduced (in terms of features, not instances) unlabeled sets, over which our active sampling method can be applied. Once the final reduced training set is created by the union of the documents selected in each partition, we apply the supervised on-demand association rule algorithm to rank the test set. One of the key advantages of our method is that it can be directly applied on an unlabeled set containing the extracted features for the documents in the corpus, without the need of an initial labeled set. Furthermore, once the unlabeled set is generated (i.e. feature extraction is performed on the corpus) and processed (i.e. discretized and partitioned), the method can be directly applied without extra parametrization since it naturally converges while selecting the training samples. This is different from most previous work where there is no clear stop criterion and the number of iterations has to be empirically determined.

We compare our method against a number of baselines that use the complete labeled training set, including some published by the LETOR dataset producers with many supervised L2R algorithms, and show that it is competitive when compared to several of them while selecting and using as little as 1.12% and at most 2.18% of the original training sets (average of 1.63% for all datasets considered). Best results reported in the literature with other active sampling strategies for L2R reported selecting at least 11% of the training set to produce similar competitive results. Moreover, in most cases our method improved the results of the original on-demand associative method (by as much as 13%), or produced similar results when compared to using the whole training set, confirming the hypothesis of noise reduction.

2 Related Work

Some researchers have recently proposed active learning schemes for L2R based on the optimization of approximations of position-based measures. The authors of [9], for example, propose a general active learning framework based on expected loss optimization (ELO). The framework uses function ensembles to select training examples that minimize a chosen loss function. The authors approach the unique challenges of active learning in L2R by separating their selection algorithm into query level and document level parts (what they call *Two-stage Active Learning*). To approximate their chosen metric, namely, DCG (*Discounted Cumulative Gain*), they use ensembles of learners to produce relevance scores and estimate predictive distributions for the documents in the active learning set. To produce the ensemble, they use a bootstrap technique that relies on an initial labeled set. Thus, their technique requires an initial labeled set to build the ensemble of learners, which needs to be large enough for the learners in the ensemble to be minimally effective. In their case, the initial labeling sets contain thousands of instances.

In [21], an SVM-specific active learning method is proposed that interactively selects the most ambiguous set of samples with respect to the learned ranking function and a initial set of “real-world queries”. At each round, those instances that minimize the support vector margin for the function learned so far are selected and the user is required to input a partial order for these instances. Compared to “traditional” active learning, this means a much more laborious labeling process for the user, who has to partially order results for every query in the initial set. This procedure is repeated an empirically established number of times with the function learned from all the user ordered instances, always selecting the most ambiguous instances for the current learned function. Although the algorithm could be modified to be used for document retrieval, in the paper it is tested using a data retrieval application which enables fuzzy search on relational databases.

Another SVM-specific strategy is presented in [2]. The authors rely on the relationship between the area under de ROC curve (AUC) and the hinge rank loss proposed by [18] to develop a loss minimization framework for active learning in ranking. Instead of testing each and every unlabeled sample to determine the one that has the smallest expected future error, the authors suggest selecting the examples that have the largest contribution to the estimated current error. These are potentially the ones that will bring more benefit when labeled for the functions that will be trained in the next rounds of the method. The proposed selection criterion is based on the hinge rank loss calculated on a per query basis and depends on the determination of a rank threshold that estimates the rank position that separates the lowest ranked relevant element from the highest ranked non-relevant example. The algorithm starts with a small labeled per query set and proceeds selecting unlabeled samples that have the highest uncertainty (as defined by the rank threshold). These samples are then labeled and added to the per query labeled sets and the process is repeated as many times as desired. The method is experimentally tested using the TD2003 and TD2004 datasets from LETOR and compared against four sampling baselines. The experimental setup starts with 11 labeled samples per query and selects 5 new samples per query per round. On the 20^{th} iteration, the selected set corresponds to approximately 11% of the original training sets in both collections.

A slightly different approach is proposed in [3]. Their method relies on the estimated risk of the ranking function on the labeled set after adding a new instance with all possible labels. The authors present results using this sampling technique with RankSVM and RankBoost. Their method, which also relies on an initial labeled training set and incrementally adds new samples, achieves competitive results with around 15% of the original training sets.

In contrast, our method does not use any initial labeled set and selects all samples from the unlabeled set. Furthermore, there is no need to empirically determine (using the validation or test sets) how many iterations are required to obtain competitive results, since the selection process naturally converges. Finally, our method achieves very good results with as little as 1.12% of the original training sets.

3 Selective Sampling Using Association Rules

In our context, the task of learning to rank is defined as follows. We have as input the *training data* (referred to as $\mathcal{D}$), which consists of a set of records of the form $< q, d, r >$, where q is a query, d is a document (represented as a list of m feature-values or $\{f_1, f_2, \ldots, f_m\}$), and r is the *relevance* of d to q. Features include BM25, Page Rank, and many other document and query-document properties. The relevance of a document draws its values from a discrete set of possibilities $\{r_0, r_1, \ldots, r_k\}$ (e.g. 0: not relevant, 1: somewhat relevant, 2: very relevant). The training data is used to build functions relating features of the documents to their corresponding relevance. The *test set* (referred to as $\mathcal{T}$) consists of records $< q, d, ? >$ for which only the query q and the document d are known, while the relevance of d to q is unknown. Ranking functions obtained from $\mathcal{D}$ are used to estimate the relevance of such documents to the corresponding queries.

3.1 Learning to Rank Using Association Rules

Ranking functions exploit the relationship between document features and relevance levels. This relationship may be represented by association rules. We denote as $\mathcal{R}$ a rule-set composed of rules of the form $\{f_j \wedge \ldots \wedge f_l \xrightarrow{\theta} r_i\}$. These rules can contain any mixture of the available features in the antecedent and a relevance level in the consequent. The strength of the association between antecedent and consequent is measured by a statistic, θ, which is known as *confidence* [1] and is simply the conditional probability of the consequent given the antecedent. In this section we discuss the use of association rules for the sake of learning to rank, and we present the algorithm LRAR (Learning to Rank using Association Rules) which extracts rules from $\mathcal{D}$ on a demand-driven basis and then combine these rules in order to estimate the relevance of each document in $\mathcal{T}$.

Demand-Driven Rule Extraction. The search space for rules is huge, and thus, computational cost restrictions must be imposed during rule extraction. Typically, a minimum support threshold (σ_{min}) is employed in order to select frequent rules (i.e., rules occurring at least σ_{min} times in $\mathcal{D}$) from which the ranking function is produced. This strategy, although simple, has some problems. If σ_{min} is set too low, a large number of rules will be extracted from $\mathcal{D}$, and often most of these rules are useless for estimating the relevance of documents in $\mathcal{T}$ (a rule $\{\mathcal{X} \rightarrow r_i\}$ is only useful to estimate the relevance of document $d \in \mathcal{T}$ if the set of features $\mathcal{X} \subseteq d$, otherwise the rule is meaningless to d). On the other hand, if σ_{min} is set too high, some important rules will not be included in $\mathcal{R}$, causing problems if some documents in $\mathcal{T}$ contain rare features (i.e., features occurring less than σ_{min} times in $\mathcal{D}$). Usually, there is no optimal value for σ_{min}, that is, there is no single value that ensures that only useful rules are included in $\mathcal{R}$, while at the same time important rules are not missed. The method to be proposed next deals with this problem by extracting rules on a demand-driven basis.

Demand-driven rule extraction is delayed until a set of documents is retrieved for a given query in $\mathcal{T}$. Then, each individual document d in $\mathcal{T}$ is used as a filter to remove irrelevant features and examples from $\mathcal{D}$. This process produces a projected training data, $\mathcal{D}_d$, which is obtained after removing all feature-values not present in d. Then, a specific rule-set, $\mathcal{R}_d$ extracted from $\mathcal{D}_d$, is produced for each document d in $\mathcal{T}$.

Lemma 1: All rules extracted from $\mathcal{D}_d$ (i.e., $\mathcal{R}_d$) are useful to estimate r.

Proof: Since all examples in $\mathcal{D}_d$ contain only feature-values that are present in d, the existence of a rule $\{\mathcal{X} \rightarrow r_i\} \in \mathcal{R}_d$, such that $\mathcal{X} \nsubseteq d$, is impossible. ■

Theorem 1: The number of rules extracted from $\mathcal{D}$ depends solely on the number of features in $\mathcal{D}_d$, no matter the value of σ_{min}.

Proof: If an arbitrary document $d \in \mathcal{T}$ contains at most l features then any rule matching d can have at most l feature-values in its antecedent. That is, for any rule $\{\mathcal{X} \rightarrow r_i\}$, such that $\mathcal{X} \subseteq d$, $|\mathcal{X}| \leq l$. Consequently, for $\sigma_{min} \approx 0$, the number of possible rules matching d is $k \times (l + \binom{l}{2} + \ldots + \binom{l}{l}) = O(2^l)$, where k is the number of distinct relevances. Thus, the number of rules extracted for all documents in $\mathcal{T}$ is $O(|\mathcal{T}| \times 2^l)$. ■

Relevance Estimation. In order to estimate the relevance of a document d, it is necessary to combine all rules in $\mathcal{R}_d$. Our strategy is to interpret $\mathcal{R}_d$ as a poll, in which each rule $\{\mathcal{X} \xrightarrow{\theta} r_i\} \in \mathcal{R}_d$ is a vote given by a set of features $\mathcal{X}$ for relevance level r_i. Votes have different weights, depending on the strength of the association they represent (i.e., θ). The weighted votes for relevance level r_i are summed and then averaged (by the total number of rules in $\mathcal{R}_d$ that predict relevance level r_i), forming the score associated with relevance r_i for document d, as shown in Equation 1 (where $\theta(\mathcal{X} \rightarrow r_i)$ is the value θ assumes for rule $\{\mathcal{X} \rightarrow r_i\}$):

$$s(d, r_i) = \frac{\sum \theta(\mathcal{X} \rightarrow r_i)}{|\ \mathcal{R}_d\ |}, \text{ where } \mathcal{X} \subseteq d \tag{1}$$

Therefore, for a document d, the score associated with relevance r_i is given by the average θ values of the rules in $\mathcal{R}_d$ predicting r_i. The likelihood of d having a relevance level r_i is obtained by normalizing the scores, as expressed by $\hat{p}(r_i|d)$, shown in Equation 2:

$$\hat{p}(r_i|d) = \frac{s(d, r_i)}{\sum\limits_{j=0}^{k} s(d, r_j)} \tag{2}$$

Finally, the relevance of document d is estimated by a linear combination of the likelihoods associated with each relevance level, as expressed by the ranking function $rank(d)$, which is shown in Equation 3:

$$rank(d) = \sum_{i=0}^{k} \big(r_i \times \hat{p}(r_i|d)\big) \tag{3}$$

The value of $rank(d)$ is an estimate of the true relevance of document d (i.e., r) using $\hat{p}(r_i|d)$. This estimate ranges from r_0 to r_k, where r_0 is the lowest relevance and r_k is the highest one. Relevance estimates are used to produce ranked lists of documents. All steps of LRAR are depicted in Algorithm 1.

Algorithm 1. Learning to Rank using Association Rules

```
Require: The training data $\mathcal{D}$, test set $\mathcal{T}$, and $\sigma_{min}$ ($\approx$ 0)
Ensure: $rank(d)$
for all pair $(d, q) \in \mathcal{T}$ do
    $\mathcal{D}_d \Leftarrow \mathcal{D}$ projected according to $d$
    $\mathcal{R}_d \Leftarrow$ rules extracted from $\mathcal{D}_d \mid \sigma \geq \sigma_{min}$
    for all $i \mid 0 \leq i \leq k$ do
        $s(d, r_i) \Leftarrow \frac{\sum \theta(\mathcal{X} \rightarrow r_i)}{|\mathcal{R}_d|}$
    end for
    $rank(d) \Leftarrow 0$
    for all $i \mid 0 \leq i \leq k$ do
        $rank(d) \Leftarrow rank(d) + r_i \times \hat{p}(r_i|d)$
    end for
end for
```

3.2 Rule-Based Active Sampling

In this section we present a novel algorithm referred to as SSAR (Selective Sampling using Association Rules), which relies on an effective selective sampling strategy in order to deal with the high cost of labeling large amounts of examples. The key idea of SSAR is that it may provide results as effective (or even better) as LRAR by carefully choosing a much smaller set of training examples from which it learns the ranking functions. As we will see in detail below, SSAR takes each unlabeled document in turn, obtaining its projection from the current reduced training set and generating the rules that would be used to rank the document. After it has the rules for all unlabeled documents, it chooses the one that generated the fewest rules, obtaining its label and inserting it into the current selected and labeled training set. The goal is to label as few instances as possible, while providing equal or even improved ranking performance.

Sampling Function. Consider a large set of unlabeled documents $\mathcal{U}$=$\{u_1, u_2, \ldots, u_n\}$. The problem we investigate in this section is how to select a small subset of documents in $\mathcal{U}$, such that the selected documents carry almost the same information of all documents in $\mathcal{U}$. These highly informative documents will compose the training data $\mathcal{D}$, and, ideally, $|\mathcal{D}| \ll |\mathcal{U}|$. Particularly, SSAR exploits the redundancy in feature-space that exists between different documents in $\mathcal{U}$. That is, many documents in $\mathcal{U}$ may share some of their feature-values, and SSAR uses this fact to perform an effective selective sampling strategy.

Intuitively, if a document $u_i \in \mathcal{U}$ is inserted into $\mathcal{D}$, then the number of useful rules for documents in $\mathcal{U}$ that share feature-values with u_i will possibly increase. In contrast, the number of useful rules for those documents in $\mathcal{U}$ that do not share any feature-value with u_i will clearly remain unchanged. Therefore, the number of rules extracted for each document in $\mathcal{U}$ can be used as an approximation of the amount of redundant information between documents already in $\mathcal{D}$ and documents in $\mathcal{U}$. The sampling function employed by SSAR exploits this key idea, by selecting documents that contribute primarily with non-redundant information, and these informative documents are those likely to demand the fewer number of rules from $\mathcal{D}$. More specifically, the sampling function $\gamma(\mathcal{U})$ returns a document in $\mathcal{U}$ according to Equation 4:

$$\gamma(\mathcal{U}) = \{u_i \text{ such that } \forall u_j : \ |\mathcal{R}_{u_i}| \ \leq \ |\mathcal{R}_{u_j}|\} \tag{4}$$

The document returned by the sampling function is inserted into $\mathcal{D}$, but it also remains in $\mathcal{U}$. In the next round of SSAR, the sampling function is executed again, but the number of rules extracted from $\mathcal{D}$ for each document in $\mathcal{U}$ is likely to change due to the document recently inserted into $\mathcal{D}$. The intuition behind choosing the document which demands the fewest rules is that such document should share less feature-values with documents that were already inserted into $\mathcal{D}$. That is, if only few rules are extracted for a document u_i, then this is evidence that $\mathcal{D}$ does not contain documents that are similar to u_i, and, thus, the information provided by document u_i is not redundant and u_i is a highly informative document. This simple heuristic works in a fine-grained level of feature-values trying to maximize the diversity in the training set. The extracted rules capture the co-occurrence of feature-values, helping in our goal of increasing diversity, since in this case, the document which demands the fewest rules is exactly the one which shares the least possible number of feature-values with documents already in the training data. In the case of a tie, the algorithm selects the document based on the size of the projection.

Notice that initially $\mathcal{D}$ is empty, and thus SSAR cannot extract any rules from $\mathcal{D}$. The first document to be labeled and inserted into $\mathcal{D}$ is selected from the set of available documents $\mathcal{U}$. In order to maximize the initial coverage of $\mathcal{D}$, the selected document is the one that maximizes the size of the projected data in $\mathcal{U}$, that is, it is the document d for which $\mathcal{U}_d$ is the largest. This is the document that shares more feature-values with the other documents of the collection and can be considered as the best representative of it. After the first document is selected and labeled, the algorithm proceeds using the fewest rules heuristic, as described above.

Natural Stop Condition. After selecting the first document and at each posterior round, SSAR executes the sampling function and a new example is inserted into $\mathcal{D}$. At iteration i, the selected document is denoted as $\gamma_i(\mathcal{U})$, and it is likely to be as dissimilar as possible from the documents already in $\mathcal{D}$=$\{\gamma_{i-1}(\mathcal{U}), \gamma_{i-2}(\mathcal{U}), \ldots, \gamma_1(\mathcal{U})\}$. The algorithm keeps inserting documents into the training data, until the stop criterion is achieved.

Lemma 2: If $\gamma_i(\mathcal{U}) \in \mathcal{D}$ then $\gamma_i(\mathcal{U})$=$\gamma_j(\mathcal{U})$ $\forall j$>i.

Proof: If $\gamma_i(\mathcal{U}) \in \mathcal{D}$ then the inclusion of $\gamma_i(\mathcal{U})$ does not change $\mathcal{D}$. As a result, any further execution of the sampling function must return the same document returned by $\gamma_i(\mathcal{U})$, and $\mathcal{D}$ will never change. ■

The algorithm stops when all available documents in $\mathcal{U}$ are less informative than any document already inserted into $\mathcal{D}$. This occurs exactly when SSAR selects a document which is already in $\mathcal{D}$. According to Lemma 2, when this condition is reached, SSAR will keep selecting the same document over and over again. At this point, the training data $\mathcal{D}$ contains the most informative documents, and LRAR can be applied to estimate the relevance of documents in $\mathcal{T}$. All steps of SSAR are shown in Algorithm 2.

Algorithm 2. Selective Sampling using Association Rules

```
Require: Unlabeled data $\mathcal{U}$, and $\sigma_{min}$ ($\approx$ 0)
Ensure: The training data $\mathcal{D}$
continue
    for all document $u_i \in \mathcal{U}$ do
        $\mathcal{D}_{u_i} \Leftarrow \mathcal{D}$ projected according to $u_i$
        $\mathcal{R}_{u_i} \Leftarrow$ rules extracted from $\mathcal{D}_{u_i} \mid \sigma \geq \sigma_{min}$
    end for
    if $\mathcal{D} = \emptyset$ then
        $\gamma_i(\mathcal{U}) \Leftarrow u_i$ such that $\forall u_j : |\mathcal{U}_{u_i}| \geq |\mathcal{U}_{u_j}|\}$
    else
        $\gamma_i(\mathcal{U}) \Leftarrow u_i$ such that $\forall u_j : |\mathcal{R}_{u_i}| \leq |\mathcal{R}_{u_j}|\}$
    end if
    if $\gamma_i(\mathcal{U}) \in \mathcal{D}$ then break
    else append $\gamma_i(\mathcal{U})$ to $\mathcal{D}$
```

4 Experimental Evaluation

4.1 LETOR Datasets

To evaluate the effectiveness of our method, we did extensive experimentation using the Learning to Rank (LETOR) benchmark datasets version 3.0. LETOR 3.0 is composed of 6 separate web datasets plus the OHSUMED corpus. The web datasets contain labeled instances selected from web pages obtained from a 2002 crawl of the .gov TLD. These collections are separated in three search tasks: topic distillation (TD), homepage finding (HP) and named page finding (NP) and contain 2 sets each (namely, TREC2003 and TREC2004). The collections contain instances represented by 64 features for the top 1.000 documents returned for a specific set of queries using the BM25 model [12]. These datasets use a binary relevance judgment indicating whether a document is or is not relevant to a given query. We evaluate our method on all the largest, more diverse, LETOR 3.0 web datasets. In all datasets we have used the query-based normalized versions as suggested by the producers of the LETOR benchmarking datasets. We also use 5-fold cross validation for all results reported as well as the evaluation script provided in the LETOR package to generate the final precision, MAP and NDCG metrics.

4.2 Results

As described in Section 3, SSAR, our rule-based active sampling algorithm processes a given list of unlabeled instances selecting the one that produces the fewest rules. In this setup, the training set for the selection process is initially empty and grows one instance at a time as they are selected from the unlabeled set and labeled. The algorithm eventually converges when it selects an instance it has already selected before. Once that happens, the selected items can be used as a reduced training set to rank the test set using LRAR (i.e. the supervised rule-based rank-learner). All results presented here use, therefore, 2 distinct sets. The original training set is used as the unlabeled set from which instances are selected by SSAR. The test set is then ranked by the LRAR using

Table 1. SSAR MAP Results and Statistics

	SSAR	LRAR	LBM25	Random	G%	Sel	Utot	Sel%	Rsel%	R%	R25%
TD2003	0.2032	0.2459	0.1402	0.1181±0.0234	44.94	157	29,435	0.53	6.80	0.82	13.22
TD2004	0.1792	0.2463	0.1452	0.1267±0.0163	23.47	141	44, 488	0.32	7.82	1.50	11.32
NP2003	0.7202	0.6373	0.5346	0.4981±0.0688	34.72	207	89, 194	0.23	3.04	0.10	25.32
NP2004	0.4993	0.5155	0.2672	0.3695±0.0575	35.15	181	44, 300	0.41	3.18	0.10	5.76
HP2003	0.6487	0.7083	0.5230	0.5486±0.0493	18.25	218	88, 564	0.25	3.62	0.12	26.04
HP2004	0.6332	0.5443	0.3712	0.3117±0.0496	70.55	222	44, 645	0.50	1.68	0.11	4.24

Table 2. SSAR with Partitions (SSARP) MAP Results and Statistics

	SSARP	LRAR	LBM25	Random	G%	Sel	Utot	Sel%	Rsel%	R%	R25%
TD2003	0.2689	0.2459	0.2104	0.1573±0.0198	27.84	642	29,435	2.18	4.40	0.82	7.70
TD2004	0.2006	0.2463	0.1818	0.1707±0.0112	10.37	633	44,488	1.42	5.72	1.50	6.00
NP2003	0.6960	0.6373	0.6321	0.5573±0.0423	10.09	995	89,194	1.12	2.42	0.10	7.28
NP2004	0.5499	0.5155	0.4064	0.4038±0.0580	35.29	860	44,300	1.94	1.80	0.10	2.16
HP2003	0.7411	0.7083	0.6487	0.5825±0.0460	14.25	1091	88, 564	1.23	2.00	0.12	6.90
HP2004	0.6168	0.5443	0.3685	0.3718±0.0479	65.89	855	44,645	1.91	1.68	0.11	1.74

the selected and labeled instances as training. Observe that our method does not use an initial labeled set to learn an initial model. The labeling effort is restricted to the examples selected by the algorithm directly from the unlabeled set[2].

Experimental Setup and First Results. The association rule mining algorithm uses nominal values to generate the inference rules used in ranking the results or in selecting samples. Therefore it is necessary to discretize the original LETOR data. Since all our data is unlabeled, we need to use an unsupervised discretization algorithm. Simple algorithms such as Uniform Range Discretization (URD) or Uniform Frequency Discretization (UFD) could be used, but being oversimplistic, they may cause some loss of information. Instead, we use the Tree-based Unsupervised Bin Estimator (TUBE) proposed in [14]. TUBE is a greedy non-parametric density estimation algorithm that uses the log-likelihood measure and a top-down tree building strategy to define varying-length bins for each attribute in the dataset. The algorithm can automatically determine the number of bins using cross-validation and the log-likelihood. Since this approach can lead to too many bins in some cases, we chose to use 10 varying-length bins for all attributes in all datasets (which is the default for the URD algorithm implemented in Weka). The results shown in tables 1 and 2 were obtained using TUBE to discretize each feature from the training set into 10 bins. After the bins were determined using the training sets in each fold, the test sets were discretized using the same bins.

Table 1 presents the results for this initial setup. The first column, “SSAR”, shows the MAP obtained using only the samples selected from the unlabeled set. The number of selected samples appears in the “Sel” column. The “UTot” column shows the number

[2] In our experiments, the presence of the user who would provide the labels is simulated; we use instead the original labels available in the collection after a document is selected.

of instances in the unlabeled set (from which the samples were selected) and the "Sel%" column indicates the percentage of the unlabeled set that the selected examples represent. The "RSel %" column shows the proportion of relevant samples in the selected set while the "R%" field shows the proportion of relevant documents in the full training set. "R25%" shows the proportion of relevant samples selected by the BM25 baseline described below. As a reference result, the "LRAR" column contains the MAP obtained by the LRAR algorithm using the complete training set and supervised discretization. We show this result for comparison, since it uses a very effective supervised discretization method [5] and provides a target to measure our hypothesis of noise reduction. The baselines appear on the 3^{rd} and 4^{th} columns: "LBM25" shows the resulting MAP from running LRAR with the same amount of samples selected by SSAR as training but selected using the value of the BM25 attribute in descending order (i.e. instances with the highest BM25 were used); "Random" shows the MAP obtained by randomly selecting the same amount of samples selected by SSAR and using these as the training set for supervised LRAR ranking. Finally, the "G%" column indicates the gain obtained by SSAR over the best value from the LRAR BM25 and Random baselines. The Random baseline was produced by randomly sampling the same amount of instances selected by SSAR at least 20 times for each fold and averaging the resulting MAPs. We present the mean obtained from all runs and all folds as well as the confidence interval for a confidence level of 95%. The LRAR with BM25 baseline tries to select more interesting instances by using the BM25 attribute value as a measure of instance quality.

From Table 1 we can see that the method converges very fast, selecting from 0.23 to 0.53% of the instances in the unlabeled set. Compared to the baselines, it performs very well as can be seen on the "G%" column. SSAR even beats the LRAR reference result in NP2003 and HP2004. Notice also that SSAR tends to select a much higher proportion of relevant instances than present in the original sets (RSel% vs. R%). These datasets have an extremely reduced amount of relevant instances and SSAR seems to single them out based on the "fewer rules" heuristic. On the other hand, the BM25-based selection obtains an even higher proportion of relevant documents, showing the power of this classic IR measure. But from the LBM25 results we can see that it is not only a matter of using many relevant instances, but also about the "quality" of the instances used. Our rule-based selection method does choose many relevant instances, but it does so based on the rules created from all attributes. These initial results are promising, but the algorithm is converging too fast in some collections for the selected samples to have sufficient representativeness and diversity. Next we propose a simple and yet effective strategy to delay the convergence.

Increasing Sample Diversity. The approach described above has two potential issues: first, it may take some time to run on datasets with too many features, since the active sampling algorithm's complexity depends on the number of features and the size of the unlabeled set. Second, as we can see in Table 1, the number of instances selected using this method is very small, ranging from 0.23 to 0.53% of the unlabeled set size. There's no doubt that the selected samples are very informative, since the results obtained by SSAR are usually reasonably better than the baselines. Nonetheless, the resulting training dataset may still be too small to provide, in some collections, the minimum diversity of examples for the supervised algorithm to reach a performance comparable to that obtained using the complete training set (i.e. column “LRAR” in Table 1). By delaying convergence and increasing the number of sampled instances we can improve the diversity of the selected set.

We propose partitioning the unlabeled set into vertical sections containing subsets of the features[3]. Each of these partitions contain all unlabeled instances, but only a group of each instance’s features. The active sampling algorithm can then be run on each partition, selecting instances based on distinct feature sets. This simple strategy increases the number of selected instances, since for each partition the algorithm will choose those that are most informative given the features in that partition. It also improves the diversity of the samples, as they are selected based on distinct characteristics. To apply this strategy, we need to determine how to separate the features into the partitions and how many partitions should be used. The features need to be divided in such a way as to allow the algorithm to select informative samples regardless of which feature set is used. However, features have diverse informational values, with some being more informative to the rank learning algorithm than others. Ideally, we want to distribute the most informative features through all partitions in order to maximize the quality and variety of the instances selected by SSAR from each partition. There are many ways to estimate which features provide more information. We chose to estimate the χ^2 value of each feature. Since we do not have relevance information, we cannot calculate the χ^2 in relation to the label. What we do instead is to calculate the χ^2 of each feature in relation to the others (i.e. how well one feature performs in predicting another feature’s value). Thus we produce a $n \times n$ matrix containing in line i the $n-1$ features ranked in descending order of their χ^2 value in relation to the i_{th} feature. We then combine the n rankings by scoring each feature according to its positions in all rankings. If a feature appears at the first position in a ranking we add 1 to its score, otherwise, we add $1/log(10 \times j)$ where j is the feature’s position in that ranking. Then we order the features in descending order of score, obtaining the final ranking.

With this ranked list of features, we can now vertically partition the unlabeled set by spreading the features into the partitions in the ranked order. Thus, given the ranked list of m features $\mathcal{F} = \{f_1, f_2, ..., f_m\}$, and a set of n partitions $\mathcal{P} = \{\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_n\}$, we place feature f_1 (the one in the first position of the χ^2 ranking) in partition $\mathcal{U}_1$ then f_2 in $\mathcal{U}_2$, eventually reaching partition $\mathcal{U}_n$ and starting over by placing the next feature, f_i into $\mathcal{U}_1$ and f_{i+1} into $\mathcal{U}_2$ and so on. We now run SSAR using each $\mathcal{U}_k$ set as input, and obtaining n sets of selected documents $\mathcal{D}_k$. We then obtain the original set of features for all selected documents from $\mathcal{U}$ and run LRAR using this new set as training.

The next step is to determine how many partitions to use. If very few partitions are used, then the increase in document diversity and the number of selected instances may be small, yielding only marginal gains. If we use too many partitions, then the informational value of each document is diluted and SSARP (SSAR with Partitions) will select many instances that are not informative as a whole, but only when considered in the context of the reduced feature set (i.e. the partition from which it was selected). Thus it is important to determine how many partitions to use to obtain good diversity while selecting informative samples. We performed preliminary tests using 2 collections, and

[3] Another strategy would be to select documents *per query*, but this would entail a large number of actively selected samples, thus hurting our goal of significantly reducing the labelling effort.

Table 3. MAP for SVM using selected samples, BM25 and Random Baselines

	SSARP	**SVMS**	**SBM25**	**Random**	**G%**
TD2003	0.2689	0.2194	0.1568	0.1417±0.0285	39.95
TD2004	0.2006	0.1957	0.1335	0.1687±0.0145	16.01
NP2003	0.6960	0.6428	0.6587	0.5739±0.0237	-2.41
NP2004	0.5499	0.5929	0.5811	0.5787±0.0329	2.04
HP2003	0.7411	0.6747	0.7090	0.5798±0.0592	-4.84
HP2004	0.6168	0.6734	0.6731	0.5406±0.0357	0.05

Table 4. MAP for SSARP and LETOR Baselines

	SSARP	**RBoost**	**FRank**	**REG**
TD2003	***0.2689***	*0.2274*	*0.2031*	*0.2409*
TD2004	0.2006	0.2614	0.2388	0.2078
NP2003	**0.6960**	0.7074	*0.6640*	*0.5644*
NP2004	*0.5499*	0.5640	0.6008	*0.5142*
HP2003	***0.7411***	*0.7330*	*0.7095*	*0.4968*
HP2004	*0.6168*	0.6251	0.6817	*0.5256*
Avg.	*0.5122*	0.5197	0.5163	*0.4250*

decided to set the number of features per partition to be from around 8 to 12. Using 5 partitions for all datasets, we have around 12 features per set for LETOR 3.0[4].

Table 2 presents the results for SSARP. All results were produced by splitting the unlabeled sets in 5 partitions, selecting instances from the partitions, creating a new labeled training set comprised of all the selected samples with all features and running LRAR on the test set with 5-fold cross validation. Again we compare the resulting MAPs with two baselines: LRAR BM25 and Random. The baselines were ran again, since the number of documents selected has changed for all datasets. Notice that the results for "LRAR" are the same as in Table 1, since it uses the complete training set.

As we can see, SSARP selects from 4 to 5 times the amount of instances selected by SSAR although the percentages relative to the unlabeled set (column "Sel%") remain very low. These percentages now vary from 1.12 (NP2003) to 2.18% (TD2003). Not only the size of the selected set increased, but there is more diversity in the selected set as instances were chosen based on different feature groups. As a consequence, the MAP for SSARP is better for 4 datasets. The improvement over SSAR was 32% for TD2003, 14% for HP2003, 11% for TD2004 and 10% for NP2004. For NP2003 and HP2004, there was a small reduction of around 3%. The proportion of relevant instances selected is also lower for both SSARP and the BM25 baseline. Our method is not only still better than the baselines but also beats LRAR using the full training sets on all datasets except TD2004. This is an indication that there is some noise reduction in the active selection process. With only 2% of the original training set it is possible to obtain better results than using the full training set with a supervised method such as LRAR.

5 Discussion

5.1 Does the Proposed Sampling Technique Work with Other L2R Methods?

Once the instances are actively selected by SSARP, it is possible to use other supervised learning algorithms to rank the test sets. Of course, different algorithms will find the instances selected by SSARP more or less "informative". Active learning methods currently proposed in the literature are inherently algorithm-specific as illustrated by the

[4] For datasets with more features, more partitions should be used.

SVM-specific techniques discussed in section 2. For instance, the approach proposed in [21] selects, at each round, the samples that minimize the support vector margin. What if we run an SVM ranking algorithm using the selected instances as training[5]? Table 3 shows the MAP resulting from running SVMRank[6] [6] using the examples selected by SSARP as training sets. The column "SSARP" repeats the results from Table 2 for reference. "SVMS" shows the MAP obtained from running SVMRank using the samples selected by SSARP (see Table 2 for other information such as the number of instances selected, etc.). Again, we ran two baselines for comparison: "SBM25" contains the results from running SVMRank with the same amount of samples as selected by SSARP, but picked by their BM25 value. "Random" shows the average of 20 runs where the instances are randomly selected and includes the confidence interval for a 95% confidence level. "G%" indicates the gain of SVMS over the best baseline.

From the results we can observe that SVMS fares very well on the TD2003 and TD2004 datasets as compared to the baselines. For the other datasets, it basically ties with the SBM25 baseline. This may be due to the fact that the BM25 method selects a high proportion of relevant instances which are extremely rare in the original NP and HP datasets (column "R%" of Tables 1 and 2). This is one of the difficulties in ranking these datasets and both SSARP and the BM25 selection methods may help as they tend to select a larger proportion of relevant samples. Though these results can be improved, they illustrate that our method chooses informative instances that are useful even for completely different algorithms.

5.2 How Does the Proposed Method Fare against Supervised LETOR Baselines?

To put SSARP results in perspective, Table 4 shows the MAP results for SSARP and those for three supervised algorithms from the LETOR 3.0 published baselines (that use the full training sets). The producers of the LETOR datasets have published results containing precision, MAP and NDCG obtained using 12 L2R algorithms on the LETOR 3.0 datasets [12]. We chose to show the results for three algorithms: RankBoost ("RBoost" in Table 4), FRank and Regression ("REG"). The first two were selected mainly because, similarly to our method, they use non-linear ranking functions, so they all lie in the same class of algorithms, making this a more fair comparison. RankBoost achieves very good results, beating all other 11 algorithms in 2 of the 6 datasets (TD2004, NP2003). FRank has average results if compared to the other algorithms and Regression obtains a lower average then the other algorithms, although it still beats some of the them in some datasets. We use the results of these algorithms as high, medium and low watermarks to show that SSARP obtains competitive results selecting only a very small fraction of each dataset. In Table 4, SSARP results that appear in *italic* are equal or better than one of the 3 baselines. Numbers in **bold** indicate that the result is equal or better that 2 of the baselines. Finally, the results in ***bold and italic*** are equal or better then all three baselines. We indicate which results from the baselines were matched or surpassed by showing them in *italic*. From Table 4 we can see that SSARP beats the chosen supervised baselines in TD2003 and HP2003. It also obtains

[5] One reason to use RankSVM is that it is one of the best performing methods on LETOR 3.0.

[6] Best parameters were determined using cross-validation on the training sets.

Table 5. NDCG for SSARP and LETOR Baselines

	SSARP			**RankBoost**			**FRank**			**Regression**		
	@1	@5	@10	@1	@5	@10	@1	@5	@10	@1	@5	@10
TD2003	***0.3800***	***0.3255***	***0.3302***	*0.2800*	*0.3149*	*0.3122*	*0.3000*	*0.2468*	*0.2690*	*0.3200*	*0.2984*	*0.3263*
TD2004	*0.3600*	*0.3404*	*0.3061*	0.5067	0.3878	0.3504	0.4933	0.3629	0.3331	*0.3600*	*0.3257*	*0.3031*
NP2003	**0.5867**	***0.7850***	**0.7918**	0.6000	*0.7818*	0.8068	*0.5400*	*0.7595*	*0.7763*	*0.4467*	*0.6423*	*0.6659*
NP2004	*0.4133*	**0.6546**	*0.6805*	0.4267	*0.6512*	0.6914	0.4800	0.6870	0.7296	*0.3733*	*0.6135*	*0.6536*
HP2003	***0.7200***	**0.7809**	**0.7982**	*0.6667*	0.8034	0.8171	*0.6533*	*0.7780*	*0.7970*	*0.4200*	*0.5463*	*0.5943*
HP2004	**0.5200**	*0.6981*	*0.7111*	*0.5067*	0.7211	0.7428	0.6000	0.7486	0.7615	*0.3867*	*0.6130*	*0.6468*

very good results for NP2003, beating 2 baselines and almost reaching the performance of the 3rd (RankBoost). Overall, SSARP's average performance is only 1.4% below the best algorithm (RankBoost) but using on average 98% less training. For further performance comparisons, Table 5 provides the NDCG@1, @5 and @10 using the same baseline algorithms and markup scheme.

5.3 How Does SSARP Compare to Other Active Learning Methods for L2R?

Some of the most recent active learning strategies for L2R [2,3] were run on the TD2003 and TD2004 datasets, thus allowing some comparison with our strategy. Both works use a similar overall sampling strategy: an initial per query labeled seed set is used and the algorithms iteratively select new samples to be labeled. The initial seed sets contain 11 ([2]) or 16 ([3]) samples per query with one of the samples being necessarily relevant (amounting to 1.1 and 1.6% of the training sets for both datasets, respectively). The algorithms can be run for as many rounds as necessary, always selecting 5 new samples *per query* at each round. By the 20th round for the first article, 11% of the training sets are selected and labeled. For the second paper, by the 25th round, 15% of the training sets are selected. In both cases, there is no way to know *a priori* how many iterations are necessary for convergence. In contrast, our method does not require initial labeled sets nor do we have to empirically determine how many rounds are needed to achieve good results. For TD2003, SSARP selects only 2.18% of the original training set and achieves a higher MAP then those reported in both papers. TD2004 was the hardest collection for our method and we did not beat the results reported in [2], but SSARP uses only 1.42% of the original training set to obtain a reasonable performance, while in that work around 11% of the training data had to be actively labeled.

6 Conclusions

We have proposed a rule-based active sampling method that is practical and effective, selecting very few documents to be labeled and yet offering competitive results when compared to established supervised algorithms using the complete training sets. Since the method does not depend on an initial labeled set, it can be easily used in real-life applications where labeling many documents can be prohibitively expensive or unpractical. For future work we intend to investigate the use of other discretization techniques.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD 1993, pp. 207–216 (1993)
2. Donmez, P., Carbonell, J.G.: Active sampling for rank learning via optimizing the area under the ROC curve. In: SIGIR 2009, pp. 78–89 (2009)
3. Donmez, P., Carbonell, J.G.: Optimizing estimated loss reduction for active sampling in rank learning. In: ICML 2008, pp. 248–255 (2008)
4. Donmez, P., Carbonell, J.G., Bennett, P.N.: Dual strategy active learning. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 116–127. Springer, Heidelberg (2007)
5. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI 1993, pp. 1022–1029 (1993)
6. Joachims, T.: Optimizing search engines using clickthrough data. In: SIGKDD 2002, pp. 133–142 (2002)
7. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: SIGIR 1994, pp. 3–12 (1994)
8. Liu, T.: Learning to rank for information retrieval. Found. Trends Inf. Retr. 3(3), 225–331 (2009)
9. Long, B., Chapelle, O., Zhang, Y., Chang, Y., Zheng, Z., Tseng, B.: Active learning for ranking through expected loss optimization. In: SIGIR 2010, pp. 267–274 (2010)
10. Mccallum, A.K.: Employing EM in pool-based active learning for text classification. In: ICML 1998, pp. 350–358 (1998)
11. Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In: ICML 2004, p. 79 (2004)
12. Qin, T., Liu, T., Xu, J., Li, H.: LETOR: a benchmark collection for research on learning to rank for information retrieval. Inf. Retr. 13, 346–374 (2010)
13. Robertson, S.E., Walker, S., Hancock-Beaulie, M.M.: Large test collection experiments on an operational, interactive system: Okapi at TREC. IP&M 31, 345–360 (1995)
14. Schmidberger, G., Frank, E.: Unsupervised discretization using tree-based density estimation. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 240–251. Springer, Heidelberg (2005)
15. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison
16. Settles, B., Craven, M., Ray, S.: Multiple-instance active learning. In: Advances in Neural Information Processing Systems, vol. 20, pp. 1289–1296. MIT Press, Cambridge (2008)
17. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: COLT 1992, pp. 287–294 (1992)
18. Steck, H.: Hinge rank loss and the area under the ROC curve. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 347–358. Springer, Heidelberg (2007)
19. Veloso, A.A., Almeida, H.M., Gonçalves, M.A., Meira Jr., W.: Learning to rank at query-time using association rules. In: SIGIR 2008, pp. 267–274 (2008)
20. Wang, L., Lin, J., Metzler, D.: Learning to efficiently rank. In: SIGIR 2010, pp. 138–145 (2010)
21. Yu, H.: SVM selective sampling for ranking with application to data retrieval. In: SIGKDD 2005, pp. 354–363 (2005)

Parallel Structural Graph Clustering

Madeleine Seeland[1], Simon A. Berger[2], Alexandros Stamatakis[2],
and Stefan Kramer[1]

[1] Technische Universität München,
Institut für Informatik/I12,
85748 Garching b. München, Germany
{madeleine.seeland,stefan.kramer}@in.tum.de
[2] Heidelberg Institute for Theoretical Studies,
69118 Heidelberg, Germany
bergers@in.tum.de, Alexandros.Stamatakis@h-its.org

Abstract. We address the problem of clustering large graph databases according to scaffolds (i.e., large structural overlaps) that are shared between cluster members. In previous work, an online algorithm was proposed for this task that produces overlapping (non-disjoint) and non-exhaustive clusterings. In this paper, we parallelize this algorithm to take advantage of high-performance parallel hardware and further improve the algorithm in three ways: a refined cluster membership test based on a set abstraction of graphs, sorting graphs according to size, to avoid cluster membership tests in the first place, and the definition of a cluster representative once the cluster scaffold is unique, to avoid cluster comparisons with all cluster members. In experiments on a large database of chemical structures, we show that running times can be reduced by a large factor for one parameter setting used in previous work. For harder parameter settings, it was possible to obtain results within reasonable time for 300,000 structures, compared to 10,000 structures in previous work. This shows that structural, scaffold-based clustering of smaller libraries for virtual screening is already feasible.

1 Introduction

Structured databases in various application areas, such as chemistry, provide a rich source of data that, in many cases, contain groups of structurally similar and dissimilar objects. To detect such groups in databases of graphs, graph clustering methods have been extensively investigated over the past few years. Basically, there exist two complementary approaches to graph clustering [7]. The simpler and more established one is to calculate a vectorial representation of the graphs and use standard similarity or distance measures in combination with standard clustering algorithms. The feature vector can be composed of properties of the graph and / or of subgraph occurrences [5,12]. Methods from this category have been found to be highly efficient, but imply a loss of information with respect to

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 256–272, 2011.

the graph topology. Moreover, a problem with vectorial graph representations is that it is unclear what a good or even optimal vectorial representation is. The second approach to graph clustering is to use the structure of the graphs directly [1,6,8,9,10], e.g., by computing the maximum common subgraph (MCS) between a set of graphs. These techniques have the desirable property that the calculated similarity measure is intuitive and can be visualized easily. However, the efficiency and scalability of these methods is still an open problem.

In this paper, we address the problem of clustering large graph databases according to scaffolds, i.e., large structural overlaps that are shared among all cluster members. More precisely, we require the cluster members to share at least one common subgraph that covers a specific fraction of the graphs in the cluster. An important challenge in this endeavor is the scalability to large graph data sets (of the order of 10^5 to 10^6 graphs). Graph databases such as the ones representing chemical compounds routinely encompass several hundred thousand graphs; thus, clustering methods that are able to explore and structure the vast graph space are highly desirable. Clustering large databases has emerged as a challenging research area with a large variety of applications, such as in the field of virtual screening, where the task is to analyze large databases of chemical compounds to identify possible drug candidates. By applying clustering techniques it is, for example, possible to prestructure the chemical space, e.g., for local modeling to capture the multi-mechanistic nature of many endpoints, the rediscovery of analog series or visualization. The majority of structural (i.e., scaffold-based) graph-based clustering algorithms, involving e.g., the computation of the MCS, is hardly suitable for such data sets. Graph data sets covered in related papers typically contain only several hundred graphs [1,4,6], and hardly any effort has been spent on characterizing the performance of the clustering algorithms. Only recently, a scaffold-based structural graph clustering algorithm [8] has been shown to handle graph data sets of at least 10,000 graphs. As this algorithm is still limited in performance, we present a parallel, scalable version of this algorithm in this paper. The algorithm, called PSCG (parallel structural clustering of graphs) in the following, is based on the idea of task partitioning in conjunction with refined cluster membership tests. More precisely, we used a set abstraction of graphs and a size-based clustering criterion to reduce the number of expensive subgraph search computations, which are not affordable exhaustively on large databases. Moreover, to avoid cluster comparisons with all cluster members, which grow computationally more expensive with increasing cluster size, we define a cluster representative for each cluster once a unique cluster scaffold is found.

The remainder of the paper is organized as follows: In Section 2, we present a few basic concepts and the sequential algorithm on which PSCG is based. In Section 3, we describe PSCG in detail. Section 4 presents a description of the data sets and experiments as well as an interpretation of the results. In Section 5, we give a conclusion.

2 Background

2.1 Notation and Definitions

In the following, all graphs are assumed to be labeled, undirected graphs. To be more precise, a graph and its subgraphs are defined as follows: A labeled *graph* is represented as a 4-tuple $g = (V, E, \alpha, \beta)$, where V is a set of vertices and $E \subseteq V \times V$ is a set of edges representing connections between all or some of the vertices in V. $\alpha : V \rightarrow L$ is a mapping that assigns labels to the vertices, and $\beta : V \times V \rightarrow L$ is a mapping that assigns labels to the edges. Given two labeled graphs $g = (V, E, \alpha, \beta)$ and $g' = (V', E', \alpha', \beta')$, g' is a *subgraph* of g, $(g' \subseteq g)$ if:

- $V' \subseteq V$
- $E' \subseteq E$
- $\forall x \in V' : \alpha'(x) = \alpha(x)$
- $\forall (x, y) \in V' \times V' : \beta'((x, y)) = \beta((x, y))$

Given two arbitrary labeled graphs $g_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $g_2 = (V_2, E_2, \alpha_2, \beta_2)$, a *common subgraph* of g_1 and g_2, $cs(g_1, g_2)$, is a graph $g = (V, E, \alpha, \beta)$ such that there exists a *subgraph isomorphism* from g to g_1 and from g to g_2. This can be generalized to sets of graphs. The set of common subgraphs of a set of graphs $\{g_1, ..., g_n\}$ is then denoted by $cs(\{g_1, ..., g_n\})$. Moreover, given two graphs g_1 and g_2, a graph g is called a *maximum common subgraph* of g_1 and g_2 if g is a common subgraph of g_1 and g_2 and there exists no other common subgraph of g_1 and g_2 that has more vertices than g. Finally, we define the size of a graph as the number of its vertices, i.e., $|V|$.

2.2 Problem Definition

Structural clustering is the problem of finding groups of graphs sharing some structural similarity. Instances with similar graph structures are expected to be in the same cluster provided that the common subgraphs match to a satisfactory extent. Only connected subgraphs are considered as common subgraphs. The similarity between graphs is defined with respect to some user-defined size threshold. The threshold is set such that the common subgraphs shared among a query graph and all cluster instances make up at least a certain proportion of the size of each graph. A graph is assigned to a cluster provided that there exists at least one such common subgraph whose size is equal or bigger than the threshold. In this way, an object can simultaneously belong to multiple clusters (overlapping clustering) if the size of at least one common subgraph with these clusters is equal or bigger than the threshold. If an object does not share a common subgraph with any cluster that meets the threshold, this object is not included in any cluster (non-exhaustive clustering). Figure 1 provides a sample clustering output for a data set of molecular graphs. The figure illustrates the overlapping and non-exhaustive character of the structural clustering algorithm.

Formally, we frame the problem of structural clustering as follows. Given a set of graph objects $X = \{x_1, ..., x_n\}$, we need to assign them into clusters which may

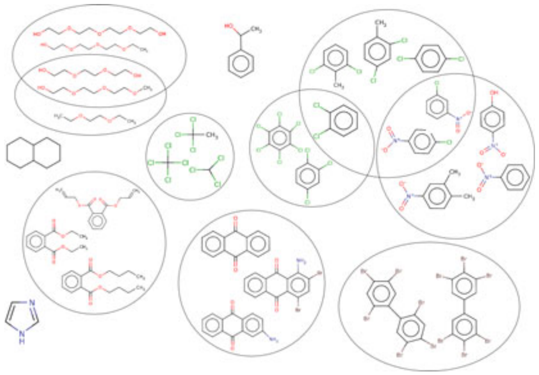


Fig. 1. Example output of PSCG on a subset of the RepDose database (`http://www.fraunhofer-repdose.de`) for $\theta = 0.7$

overlap with each other. In clustering these objects, one objective is considered: to maximize the average number of objects contained in a cluster, such that at any time for each cluster C there exists at least one common subgraph that makes up a specific proportion, θ, of the size of each cluster member. Considering the state of a cluster $C = \{x_1, ..., x_m\}$[1] at any point in time, the criterion can formally be defined as:

$$\exists\, s \in cs(\{x_1, ..., x_m\}) \forall x_i \in C : |s| \geq \theta |x_i| \tag{1}$$

where s is a subgraph and $\theta \in [0, 1]$ is a user-defined similarity coefficient. According to this goal, a minimum threshold for the size of the common subgraphs shared by the query graph x_{m+1} and the graphs in cluster C can be defined as

$$minSize = \theta\ max(|x_{max}|, |x_{m+1}|), \tag{2}$$

where $\theta \in [0, 1]$ and x_{max} is the largest graph instance in the cluster. To obtain meaningful and interpretable results, the minimum size of a graph considered for cluster membership is further constrained by a *minGraphSize* threshold. Only graphs whose size is greater than *minGraphSize* are considered for clustering. Thus, the identification of the general cluster scaffold will not be impeded by the presence of a few graph structures whose scaffold is much smaller than the one the majority of the cluster members share. This will be especially useful in real-world applications that often contain small fragments.

2.3 Sequential Structural Clustering

The proposed parallel clustering approach PSCG extends and improves a structural graph clustering approach proposed recently [8]. In short, the algorithm works as follows. Let *minGraphSize* be the minimum threshold for the graph size

[1] In slight abuse of notation, we use the same indices as above.

and *minSize* be the minimum threshold for the size of the common subgraphs specified by the user and defined in Equation 2. In the first step, an initial cluster is created containing the first graph object that is larger than *minGraphSize*. In the following steps, each instance is compared against all existing clusters. In case the query instance meets the *minGraphSize* threshold and shares at least one common subgraph with one or more clusters that meets the cluster criterion in Equation 2, the instance is added to the respective cluster. Unlike many traditional clustering algorithms, a graph object is allowed to belong to no cluster, since it is possible that an object is not similar to any cluster. In this case, a new singleton cluster is created containing the query graph instance.

For computing common subgraphs, a modified version of the graph mining algorithm gSpan [11] that mines frequent subgraphs in a database of graphs satisfying a given minimum frequency constraint is used. The structural clustering approach requires a minimum support threshold of $minSup = 100\%$ in a set of graphs, i.e., all common subgraphs have to be embedded in all cluster members. For experiments with molecular graph data, gSpan$'$, an optimization of gSpan for mining molecular databases (`http://wwwkramer.in.tum.de/projects/gSpan.tgz`) is used. As the only interest lies in the determination of at least one common subgraph that meets the minimum size threshold defined in Equation 2, the graph mining algorithm gSpan [11] was modified, to mine frequent subgraphs with a maximum size of *minSize*. More specifically, once the size of the current subgraph reaches *minSize*, it will not be grown any more and search terminates. In this way, the computation of all frequent common subgraphs can be avoided, thus achieving a substantial performance improvement.

3 Parallel Structural Graph Clustering

In this section, we present enhancements and optimizations of the structural clustering algorithm proposed by Seeland *et al.* [8] that enable PSCG to handle large data sets. The main idea of PSCG is to partition the clustering task into independent tasks which are distributed among a set of processes, i.e., each process is responsible for one cluster. The motivation behind partitioning the set of clusters instead of the graph data set is that each process can compare all relevant graph objects, i.e., all graph objects with an index greater than the index of the graph that initiated the singleton cluster, against the assigned cluster without the need to wait for the intermediate results of the other processes. To achieve this, we need a master process which is responsible for managing the cluster results of all processes.

We adopt the *master-worker paradigm* to implement PSCG. The master-worker programming model consists of two kinds of entities: a single master and multiple workers. The master is responsible for decomposing a clustering problem into a subset of clustering tasks and distributing these tasks among a farm of workers (by putting the tasks in a shared queue), as well as for gathering the partial results in order to produce the final computation result. A queue, shared between the master and the workers, is used to represent the shared space

Algorithm 1. Master

```
 1: stable_sort($graph$[]) //see Section 3.2
 2: for ($i \leftarrow 0, num\_procs - 1$) do
 3:     w ← new Worker()
 4:     w.start()
 5: end for
 6: $first = 0$
 7: while $|graph[first]| < minGraphSize$ do
 8:     $first++$;
 9: end while
10: $c \leftarrow new\ Cluster(graph[first])$
11: queue.add($c$)
12: for ($i \leftarrow first + 1, |graph[]| - 1$) do
13:     $graph[i].nrClusterComparisons++$
14: end for
15: while (true) do
16:     if (!workers.active && queue.isEmpty) then
17:         for ($i \leftarrow 0, num\_procs - 1$) do
18:             w.terminate()
19:         end for
20:         break
21:     end if
22: end while
```

where the pending clusters reside. Each worker is responsible for only one cluster at any point in time, independently computing one iteration: It pulls a clustering task (input) from the queue, processes the task by comparing all relevant graphs in the graph database against the cluster, and sends the result, i.e., the processed cluster, back to the master (output).

One of the advantages of using this pattern is that the algorithm is based on a dynamic load balancing of the cluster queue, i.e., the algorithm automatically balances the load. This is possible due to the adoption of a receiver-initiated dynamic load balancing approach based on polling: the work set is shared, and the workers continue to pull work from the set until there is no more work to be done. A static load balancing policy is not adequate for our algorithm as the work load is not known in advance and cannot be estimated easily.

In the following sections, we describe the parallel structural clustering algorithm PSCG in more detail.

3.1 Cluster Comparisons

Let *minGraphSize* be the minimum threshold for the graph size and *minSize* be the minimum threshold for the size of the common subgraphs specified by the user and defined in Equation 2. At the beginning of algorithmic execution, we start with an empty set of clusters. In the first step, the master initiates the computation by creating an initial cluster containing the first graph object that is larger than *minGraphSize* (Algorithm 1, line 6-10). The master process is

Algorithm 2. Worker

```
1: while (!terminationSignal) do
2:     $c \leftarrow queue.getCluster()$
3:     if (c != null) then
4:         PSCG($c, graphStartIdx, \theta, minGraphSize$)
5:     end if
6: end while
7: w.terminate()
```

Algorithm 3. Structural Clustering

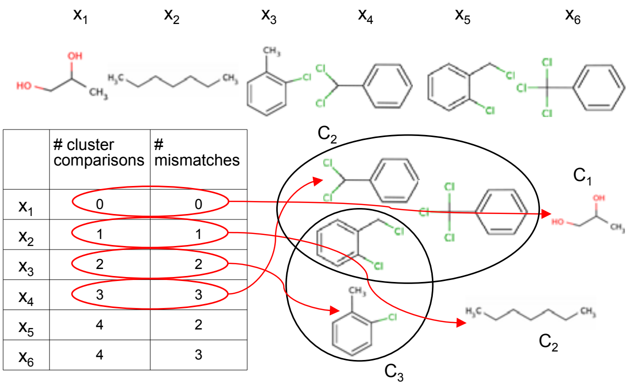
```
1: procedure PSCG($c$,$graphStartIdx$,$\theta$, $minGraphSize$)
2:     $graphEndIdx \leftarrow idx(graph : |graph| \leq \theta \cdot c.min)]$ //see Section 3.2
3:     for ($j \leftarrow graphStartIdx, graphEndIdx$) do
4:         if ($graph[j] \geq minGraphSize$) then
5:             $hasCluster \leftarrow false$
6:             if ($s(\boldsymbol{f}_{graph[j]}, \boldsymbol{f}_c) < \theta\, max(|graph[j]|, |c.min|)$) then //see Section 3.3
7:                 MISMATCH($c.id, j, j+1$)
8:                 $continue$
9:             else
10:                $minSize \leftarrow \theta \cdot max(|graph[j]|, |c.max|)$
11:                if (!UniqueScaffold) then //see Section 3.4
12:                    $minSup \leftarrow |c| + 1$
13:                    $ret \leftarrow gSpan'''(graph[j] \cup c.graphs, minSup, minSize)$
14:                else
15:                    $minSup \leftarrow 2$
16:                    $ret \leftarrow gSpan''(graph[j] \cup c.scaffold, minSup, minSize)$
17:                end if
18:                if ($ret = 1$) then
19:                    $c[last + 1] \leftarrow graph[j]$
20:                    $hasCluster \leftarrow true$
21:                end if
22:            end if
23:            if ($hasCluster = false$) then
24:                MISMATCH($c.id, j, j+1$)
25:            end if
26:        end if
27:    end for
28:    if ($graphEndIdx + 1 < |graph[]|$)) then
29:        MISMATCH($c.id, graphEndIdx + 1, |graph[]|$)
30:    end if
31:    $results.add(c)$
32: end procedure
```

responsible for putting the initial cluster in the cluster queue (line 11) which stores cluster objects that are exchanged with the workers. Subsequently, the master increases the number of necessary cluster comparisons for all subsequent graphs (explained in more detail later in this section) (line 12-13). In the fol-

Algorithm 4. Maintainance of Cluster Membership Information

```
procedure MISMATCH($cId$,$startId$,$endId$)
    for ($graphId \leftarrow startId, endId - 1$) do
        $graph[graphId].nrMismatches + +$
        if ($graph[graphId].nrCluComp = graph[graphId].nrMism$) then
            $c \leftarrow new\ Cluster(graph[graphId])$
            queue.add(c)
            for ($i \leftarrow graphId + 1, |graph[]| - 1$) do
                $graph[i].nrClusterComparisons + +$
            end for
        end if
    end for
end procedure
```

lowing steps, idle workers continue to pull one cluster at a time from the queue (Algorithm 2, line 2) and perform clustering (line 4) by comparing all graph instances in the graph database that lie within a specified index range (which will be explained in more detail in Section 3.2) against the assigned cluster (Algorithm 3, line 3). In case a query instance meets the *minGraphSize* threshold and shares at least one common subgraph with the cluster that meets the cluster criterion in Equation 2 (line 18), the instance is added to the respective cluster (line 19). In case a graph object does not belong to any cluster, a new cluster is created. In contrast to the sequential clustering setting, however, in the parallel setting the information whether a graph belongs to a cluster is distributed over the set of workers. Since a new cluster can only be created if it is not assigned to any existing cluster, the master needs to maintain the cluster membership information for all graph instances. In particular, for each graph we need to maintain two cluster membership parameters: the number of necessary cluster comparisons as well as the numbers of clusters the graph does not fit into (denoted as



	# cluster comparisons	# mismatches
x_1	0	0
x_2	1	1
x_3	2	2
x_4	3	3
x_5	4	2
x_6	4	3

Fig. 2. Graphical illustration of PSCG on a sample data set ($\theta = 0.8$)

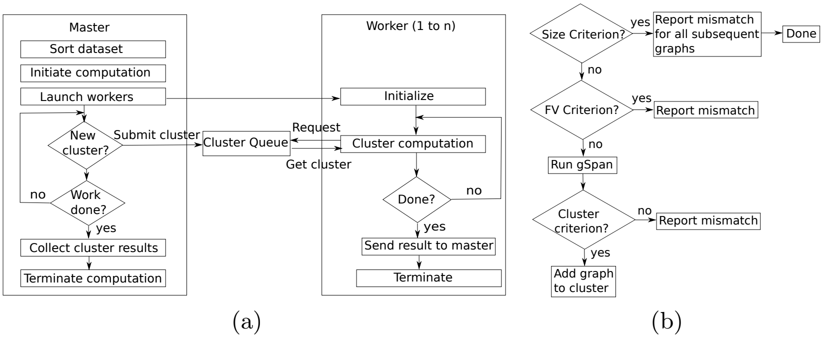


Fig. 3. Flow charts of the parallel structural clustering algorithm PSCG

the number of cluster mismatches). If a graph does not belong to a cluster the worker forwards the non-membership information to the master (Algorithm 3, line 24). Note, that due to the overlapping nature of the clustering algorithm, a graph can be directly assigned to a cluster in case it meets the cluster criterion without informing the master. Each time a worker reports a cluster mismatch for a graph, the master first increases the mismatch parameter for the graph (Algorithm 4, line 2-3) and then checks the two cluster membership parameters. If the number of necessary cluster comparisons is equal to the number of cluster mismatches (line 4), suggesting that the corresponding graph does not belong to any cluster, a new cluster is created (line 5). The master puts the cluster in the task queue (line 6) and increases the cluster comparison parameter for all subsequent graphs in the graph data set (line 7-8). A graphical illustration of the clustering process on a sample data set of molecular graphs is shown in Figure 2, where large circles represent clusters and the single structures outside denote singleton clusters. The table contains the cluster membership parameters maintained by the master. Once a worker is done with an iteration, the resulting cluster is added to the result queue managed by the master (Algorithm 3, line 31). Figure 3 illustrates the master-worker paradigm of PSCG in a flow chart.

As in the sequential clustering algorithm, we use gSpan$''$ for computing common subgraphs. Given that pairwise subgraph similarity computation is very expensive, it would be highly desirable to reduce the number of subgraph computations. Therefore, we introduced the following cluster exclusion criteria to avoid unnecessary calls to the gSpan$''$ algorithm in the first place: a refined cluster membership test based on node feature vectors of graphs, and a clustering exclusion criterion based on the size of graph objects which requires the graph data set to be sorted according to size. These criteria are used to perform a search space pruning on the actual clustering. The aim of search space pruning is to reduce the number of graph candidates in the database that need to undergo an expensive, full fledged graph matching process. Further, to reduce gSpan running times for larger clusters, we define a cluster representative for each cluster composed of the common cluster scaffold once this scaffold is unique and thus also minimal. In the following three subsections, we describe the employed cluster exclusion criteria and the intuition behind the definition of the cluster representative in more detail. The impact of these algorithmic improvements will be investigated in Section 4.

3.2 Size Based Exclusion Criterion

The cluster criterion defined in Equation 2 constrains the set of graphs being considered for clustering. More precisely, only graphs in a certain size range are considered for comparison with a specific cluster, i.e., graphs whose sizes lie in the range $[\lceil \theta x_{max} \rceil, \lfloor \frac{1}{\theta} x_{min} \rfloor]$, where x_{min} is the smallest and x_{max} is the largest graph instance in the cluster. The lower bound of the size range ensures that only graph instances that are equal to or larger than the minimum required size for at least one common subgraph, $minSize$, are considered for cluster membership. This is necessary since at any point in time at least one common subgraph should make up a proportion θ of the size of each cluster member. The upper bound excludes query instances that are larger than $minSize$ and thus would break up an existing cluster. Incorporating this information in the clustering process would give us the possibility to avoid comparing a cluster to the complete database.

To effectively employ the size based criterion, we sort the data set in increasing order of graph size. Thus, we do not need to compare the subsequent graphs against a cluster, once a query graph exceeds the upper bound of the size range (see Figure 3(b)). To preserve the incremental character (i.e., each graph in the graph database is only processed once by comparing it against all existing clusters) of the structural clustering algorithm [8], we need to make sure that the graph index corresponding to the lower bound is greater than the index following the index of the graph instance that initiated the assigned singleton cluster. However, due to the ordering of the data set by size, the graph index corresponding to the lower bound is always equal to or smaller than the index of the graph that initiated the singleton cluster. Thus, the graph indices that are considered for comparison against a cluster lie in the range $[idx(x_{min})+1, idx(x : |x| \leq \lfloor \frac{1}{\theta} x_{min} \rfloor)]$, where x_{min} is the smallest graph in the cluster. Due to the ordering of the data set, this graph corresponds to the graph that initiated the clustering. Figure 4 illustrates the use of the size based exlusion criterion during the clustering process on a data set of eight molecular graphs.

3.3 Exclusion Criterion Based on Node Feature Vectors

The second clustering exclusion criterion is based on a set abstraction of graphs, i.e., a numerical feature vector representing the number of node types in a graph. The underlying idea is that for two graphs the overlapping node set represents an upper bound for the size of the maximum common subgraph. Thus, given a query instance, we can skip the common subgraph computation with a cluster if the size of the overlapping node set of the query graph and the cluster representantive is smaller than $minSize$.

Formally, during the preprocessing phase of structural clustering, we represent each graph g_i by a numerical feature vector $\boldsymbol{f}_{g_i} = (f^1_{g_i}, .., f^n_{g_i})$ corresponding

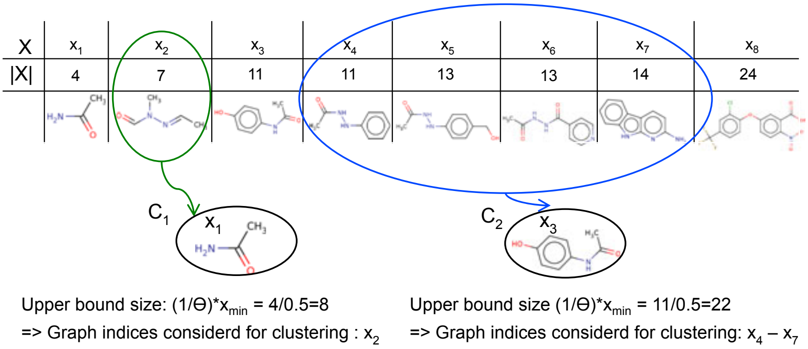

X
|X|
x1
x2
x3
x4
x5
x6
x7
x8
4
7
11
11
13
13
14
24
C1
x1
C2
x3
Upper bound size: (1/Θ)*xmin = 4/0.5=8
=> Graph indices considerd for clustering : x2
Upper bound size (1/Θ)*xmin = 11/0.5=22
=> Graph indices considerd for clustering: x4 – x7

Fig. 4. Example use of the size based cluster exclusion criterion on a data set of chemical compounds containing eight graphs ($\theta = 0.5$)

to a set of vertex types $l_1, ..., l_n$. Each entry in the feature vector records the number of a specific vertex type occuring in the respective graph. Let $f_{g_i}^k$ denote the numerical feature associated with the vertex type v_k. Each cluster C_j is represented by a vector $\boldsymbol{f}_{C_j} = (f_{C_j}^1, ..., f_{C_j}^n)$ defined in terms of the overlap of the feature vectors of the instances in that cluster, i.e., the common vertex type set shared by all cluster instances. The similarity s between $\boldsymbol{f}_{g_i}$ and $\boldsymbol{f}_{C_j}$ is computed by summing up the minimum of each pair of feature vector components

$$s(\boldsymbol{f}_{g_i}, \boldsymbol{f}_{C_j}) = \sum_k (min(\{\boldsymbol{f}_{g_i}^k \in \boldsymbol{f}_{g_i}\} \cup \{\boldsymbol{f}_{C_j}^k \in \boldsymbol{f}_{C_j}\})) \tag{3}$$

representing an upper bound on the size of the maximum common subgraph (Algorithm 3, line 6). If the similarity $s(\boldsymbol{f}_{g_i}, \boldsymbol{f}_{C_j})$ is lower than the minimum threshold for the size of the common subgraphs, $minSize$ (Equation 2), i.e., $s(\boldsymbol{f}_{g_i}, \boldsymbol{f}_{C_j}) < minSize$, we omit the computation of the common subgraphs, report the cluster mismatch to the master (line 7) and continue with the next cluster comparison (line 8). In this way, we eliminate graphs with a limited degree of resemblance to the target cluster, and increase the overall speed of the algorithm. Figure 5 shows a sample application of the feature vector criterion. In this example, the query graph x_{m+1} is compared against a cluster containing two graphs. As the similarity between the node feature vector of the query graph and the cluster is lower than $minSize$, the query graph is not considered for the cluster membership test, i.e., the computation of the common subgraphs can be omitted.

3.4 Definition of a Cluster Representative

As mentioned in Section 2.3, the structural clustering algorithm limits subgraph mining to the search of one common subgraph that satisfies the minimum size threshold, $minSize$ to avoid the computation of all frequent common subgraphs. This limitation forces us to compare each query graph against all cluster members which may have a remarkable impact on the runtime of gSpan, in

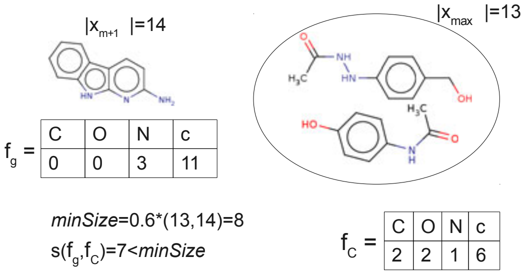


Fig. 5. Example use of the feature vector based cluster exclusion criterion ($\theta = 0.6$)

particular for larger clusters. To reduce running times, we define a cluster representative for each cluster once all cluster members share a unique cluster scaffold, i.e., the minimum required common subgraph is the only common subgraph all cluster members have in common. Since in the structural clustering algorithm [8] subgraph mining is terminated once a common subgraph is found that satisfies $minSize$, the existence of further common subgraphs is unknown. Therfore, we need to go one level deeper in the subgraph mining process and check if there exists at least another common subgraph with size equal to or greater than $minSize$. In the pseudocode, this modification of gSpan is called gSpan$'''$ (Algorithm 3, line 13). As soon as all graphs in a cluster share no more than one common subgraph, this unique subgraph is used as the cluster representative. In the following, all subsequent query graphs are compared against the cluster representative instead of comparing it against all graphs in the cluster (line 15-16). Further, subgraph mining is terminated as soon as a common subgraph of size $minSize$ is found that is covered by the query graph and the cluster representative, i.e., gSpan$''$ is used. Note, that the reason for not defining a cluster representative before the existence of a unique cluster scaffold is due to the following two reasons. First, there may exist at least another common subgraph of size $minSize$. By using the first common subgraph found as cluster representative, it may be the case that the query graph and the cluster representative share a common subgraph of size $minSize$ that is not the first common subgraph. In this case, by mistake the query graph would not be assigned to the cluster. Second, there may exist larger subgraphs. By ignoring the existence of these subgraphs and using the first common subgraph found as cluster representative, it may be the case that the $minSize$ threshold is smaller than the size of the common subgraph shared by the query graph and the cluster representative. Thus, the query graph would not be assigned to the cluster even if there exist larger common subgraphs that fulfill the size threshold.

4 Experimental Results

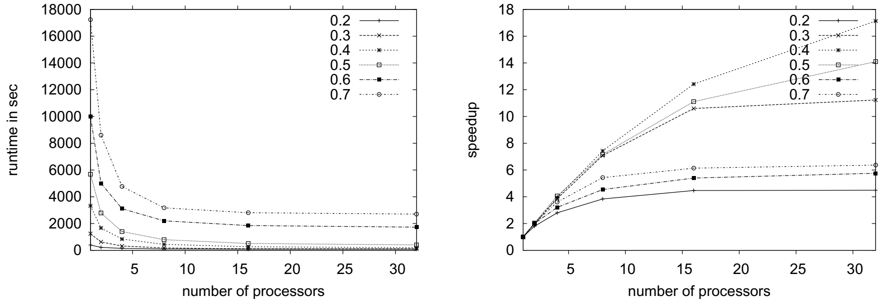
To evaluate the efficiency of our parallel structural clustering algorithm PSCG, introduced in Section 3, we conducted several experiments on several publicly available data sets of molecular graphs. In this section, we describe the data sets, the experimental set-up and the results.

4.1 Test Environment and Data Sets

The clusterings on the data sets containing up to 200,000 graphs were carried out on a SUN x4600 system with 32 AMD Opteron CPU cores (8 CPU sockets with 4 cpu cores) using the multi-threaded version of the algorithm. The processor in each node runs at 2.5 GHz with 2 GB of main memory. The clusterings on the data set containing 300,000 structures were carried out using the MPI parallelized version of the algorithm. Here, the compute cluster consists of 2016 AMD Opteron (Magny-Cours) CPU cores (42 Dell R815 nodes with 48 cpu cores and 128-256 GB main memory) and Qlogic infiniband interconnects. The algorithm was implemented in C++ using the boost libraries (`www.boost.org`) for multi-threading support. For the experiments, we employed the chemical domain as our application area by using real data sets of molecular graphs. The first data set contains the first 10,000 structures of the NCI anti-HIV database (`http://dtp.nci.nih.gov/docs/aids/aids_data.html`) which contains 36,255 compounds. The second data set, ChemDB, contains nearly 5 M commercially available small molecules [2,3]. We created data sets sized from 100,000 to 300,000 graphs from this data set using random sampling.

4.2 Performance Evaluation

We investigated the runtime performance of PSCG for different numbers of processors (1, 2, 4, 8, 16 and 32) and different values of θ using the first 10,000 graph structures from the NCI anti-HIV database. The runtime performance of PSCG was evaluated according to the speedup factor. Speedup (S) is defined as a ratio of the time taken in running the sequential algorithm (T_s) to the time taken in running the parallel algorithm (T_p) with P processors, i.e., $S = \frac{T_s}{T_p}$.



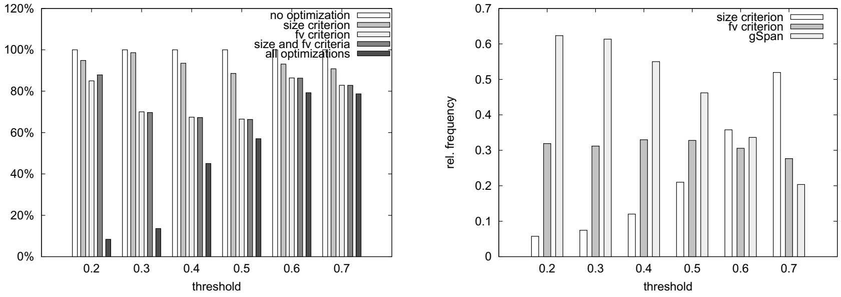
(a) (b)

Fig. 6. (a) Execution time and (b) speedup of PSCG on the first 10,000 graphs of the NCI anti-HIV data set

Figure 6 shows the execution time and the speedup for different values of θ. The results indicate that our algorithm scales well with the number of processors and has a good speedup which is close to linear for certain parameter settings, i.e., for smaller values of θ. For larger similarity coefficients, there is a higher number of computationally more demanding cluster comparisons, especially at the end of the clustering when the graphs become larger and the runtime degenerates.

4.3 Effects of Algorithm Improvements

We investigated the impact of the algorithm improvements presented in Section 3.2, 3.3 and 3.4 on the performance of PSCG. For this, we ran the algorithm on the NCI anti-HIV data set with 32 processors using (i) no optimizations, (ii) only the size based exclusion criterion, (iii) only the feature vector based exclusion criterion, (iv) both the size and feature vector based criteria and (v) all optimizations including the definition of a cluster scaffold once it is unique. Figure 7 shows the runtime reduction and an overview of the relative frequency of both exclusion criteria as well as the frequency of gSpan calls. The results indicate that significant performance improvements, especially for $\theta \leq 0.5$, can be achieved with the application of the cluster exclusion criteria and the definition of a cluster scaffold.



(a) (b)

Fig. 7. a) Runtime reduction due to algorithm improvements and b) relative frequency of size-based and feature vector based exclusion criterion and number of gSpan calls

4.4 Comparison to Sequential Structural Clustering

We compared the runtime performance of the sequential structural clustering algorithm [8] with PSCG on the first 10,000 structures of the NCI anti-HIV data set. For accurate comparison, we used the same experimental setup. We only show the experimental results for $\theta \in [0.2, 0.5]$, since for $\theta \geq 0.5$, the sequential algorithm did not terminate within a certain timeout period. Table 1 shows the runtime performance of both clustering versions. The runtime advantage of PSCG over the sequential clustering version is clear, showing improved computation efficiency by factors of 300 fold to 1900 fold for PSCG. The reasons for this can be explained by the following improvements in PSCG. First, the clustering task is partitioned into independent tasks which are distributed among a set of workers. Each worker compares the graph structures in the data set against the assigned cluster without the need to wait for the intermediate results of the other processes. Second, we introduced two clustering exclusion criteria which reduce the number of cluster membership tests. Third, we defined a cluster representative once the scaffold of a cluster is unique, to avoid cluster comparisons with all cluster members. Fourth, we reduced the invocation overhead of the individual gSpan runs. This optimization is especially efficient for gSpan runs with low overall runtimes.

Table 1. Runtime (in sec) of the sequential clustering version vs. PSCG on the first 10,000 graphs of the NCI anti-HIV data set for different values of θ

θ	0.2	0.3	0.4	0.5
t_{seq}	747,000	1,068,420	1,434,780	2,087,280
t_{par}	396	1,244	3,394	6,235

4.5 Experiments on Large Graph Data Sets

We tested PSCG on three data sets sampled from the ChemDB data set containing 100,000, 200,000 and 300,000 graphs respectively. For the experiments, we used 32 CPUs for the data sets with 100,000 and 200,000 graphs. For the data set containing 300,000 graphs we used 96 CPUs for $\theta = 0.4$. For $\theta = 0.6$, we used 96 (48) CPUs to cluster the first (second) half of the data set. The rationale for the change in the CPU number is that the parallel efficiency of the algorithm can change over the runtime of the algorithm (i.e., towards the end a large number of workers may be idle constantly). The MPI version contains a checkpoint/restart facility which allowed us to adjust the number of used CPU cores to account for this by manually balancing the workload on the cluster. Tables 2 and 3 show the runtime performance as well as the number of created clusters on the sampled data sets for $\theta = 0.4$ and $\theta = 0.6$ using all three previously described algorithmic improvements.

Table 2. Runtime (in sec) for the sampled data sets

$\|D\|$	$\theta = 0.4$	$\theta = 0.6$
100,000	31,103 •	67,563 •
200,000	122,204 •	349,568 •
300,000	610,577 ∘	1,163,761 ⋆

Table 3. Number of clusters for the sampled data sets

$\|D\|$	$\theta = 0.4$	$\theta = 0.6$
100,000	4,112	16,295
200,000	6,096	25,685
300,000	9,811	38,775

•: 32 processors ∘: 96 processors ⋆: first half: 96 processors, second half: 48 processors

5 Conclusion

In this paper, we presented PSCG, a parallel and improved version of a recently proposed structural graph clustering algorithm [8]. PSCG uses a task partitioning approach and makes use of two clustering exclusion criteria to reduce cluster membership tests. Further, to reduce gSpan running times for larger clusters, we define a cluster representative for each cluster composed of the common cluster scaffold once this scaffold is unique. To study the effectiveness of our proposed algorithm for clustering large data sets, we conducted extensive experiments. The experimental results suggest that the algorithm scales well with the increasing size of the data and, for certain parameter settings, speeds up nearly linearly with the increasing number of processors. For real world data sets, this algorithm is able to handle a much greater number of graph objects compared to previously proposed structure-based clustering algorithms. Given these performance improvements, our algorithm should already be applicable to the large structure databases from virtual screening.

References

1. Aggarwal, C.C., Ta, N., Wang, J., Feng, J., Zaki, M.: XProj: a framework for projected structural clustering of XML documents. In: KDD 2007: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 46–55. ACM, New York (2007)
2. Chen, J., Swamidass, S.J., Dou, Y., Baldi, P.: ChemDB: a public database of small molecules and related chemoinformatics resources. Bioinf. 21, 4133–4139 (2005)
3. Chen, J.H., Linstead, E., Swamidass, S.J., Wang, D., Baldi, P.: ChemDB updatefull-text search and virtual chemical space. Bioinf. 23, 2348–2351 (2007)
4. Hossain, M.S., Angryk, R.A.: GDClust: A graph-based document clustering technique. In: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops, ICDMW 2007, pp. 417–422. IEEE Computer Society, Washington, DC, USA (2007)
5. McGregor, M.J., Pallai, P.V.: Clustering of large databases of compounds: Using the MDL "keys" as structural descriptors. Journal of Chemical Information and Computer Sciences 37(3), 443–448 (1997)
6. Raymond, J.W., Blankley, C.J., Willett, P.: Comparison of chemical clustering methods using graph- and fingerprint-based similarity measures. J. Mol. Graph. Model. 21(5), 421–433 (2003)
7. Raymond, J.W., Willett, P.: Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2D chemical structure databases. J. Comput. Aided. Mol. Des. 16(1), 59–71 (2002)
8. Seeland, M., Girschick, T., Buchwald, F., Kramer, S.: Online structural graph clustering using frequent subgraph mining. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010. LNCS, vol. 6323, pp. 213–228. Springer, Heidelberg (2010)
9. Stahl, M., Mauser, H.: Database clustering with a combination of fingerprint and maximum common substructure methods. J. Chem. Inf. Model. 45, 542–548 (2005)

10. Tsuda, K., Kudo, T.: Clustering graphs by weighted substructure mining. In: ICML 2006: Proceedings of the 23rd International Conference on Machine Learning, pp. 953–960. ACM, New York (2006)
11. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining, pp. 721–724 (2002)
12. Yoshida, T., Shoda, R., Motoda, H.: Graph clustering based on structural similarity of fragments. In: Jantke, K.P., Lunzer, A., Spyratos, N., Tanaka, Y. (eds.) Federation over the Web. LNCS (LNAI), vol. 3847, pp. 97–114. Springer, Heidelberg (2006)

Aspects of Semi-supervised and Active Learning in Conditional Random Fields

Nataliya Sokolovska

LRI, CNRS UMR 8623 & INRIA Saclay,
University Paris Sud, Orsay, France
nataliya@lri.fr

Abstract. Conditional random fields are among the state-of-the art approaches to structured output prediction, and the model has been adopted for various real-world problems. The supervised classification is expensive, since it is usually expensive to produce labelled data. Unlabeled data are relatively cheap, but how to use it? Unlabeled data can be used to estimate marginal probability of observations, and we exploit this idea in our work.

Introduction of unlabeled data and of probability of observations into a purely discriminative model is a challenging task.

We consider an extrapolation of a recently proposed semi-supervised criterion to the model of conditional random fields, and show its drawbacks. We discuss alternative usage of the marginal probability and propose a pool-based active learning approach based on quota sampling. We carry out experiments on synthetic as well as on standard natural language data sets, and we show that the proposed quota sampling active learning method is efficient.

Keywords: conditional random fields, probability of observations, active learning, semi-supervised learning.

1 Introduction

In real-world applications (text, image, audio data processing) unlabeled data are plentiful and cheap. Labeled data, on the contrary, are usually rather expensive to gather. The problem how to exploit unlabeled instances is not recent and many proposals have been already made. Another problem is how to select training data. How to choose instances of high training utility is the active learning problem.

Intuitively, the information one can get from unlabeled data is the marginal probability of observations. In the asymptotic case, when we dispose infinitely many unlabeled instances, we can estimate the true marginal probability of observations. In real-world problems this is not feasible, since the number of observations is always limited. However, the probability of observations can be approximated.

Attention of the machine learning and data mining communities has been drawn to semi-supervised approaches (see [4] for an overview), especially by

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 273–288, 2011.

probabilistic semi-supervised classifiers. Logistic regression is a simple efficient discriminant model widely used for various applications. However, a number of real-world applications has sequential structure, e.g., natural language and biological applications. Conditional random fields [11] are a generalization of logistic regression, and therefore, a discriminative approach, which models sequential dependencies and allows to take a rich set of features into account.

Probabilistic generative models fare easily with unlabeled data, usually via the expectation-maximization algorithm [6,22]. Discriminative probabilistic models are reported to perform better than probabilistic generative models [17]. The introduction of unlabeled instances into discriminative models is much more challenging, since it is not straightforward how to integrate marginal probability of observations into a discriminative model.

Among the state-of-the art semi-supervised methods are combinations of generative and discriminative approaches in order to profit from both aspects, a better generalization error of a discriminative model and information extracted from unlabeled data, integrated into a generative model. A convex combination of a discriminative model and a generative model is considered e.g. by [2] and [9]. A Bayesian point of view for the hybrid approaches has been explored by [16] and [12]. The proposed hybrid method is based on the fact that parameters of a discriminative and of a generative models are related via their Bayesian distribution. However, the number of parameters to be estimated is usually doubled in the hybrid approaches, since the number of models increases.

The criterion of Bengio-Grandvalet [8] was probably the first attempt to introduce unlabeled data into a discriminant classifier. The criterion implements the idea that the classes have to be well-separated; conditional entropy over unlabeled instances is taken as a measure of overlap of classes. Among significant disadvantages of the criterion are its non-convexity and instability in cases where the number of labeled points is small.

In this paper, we discuss ideas how to introduce the marginal probability of observations into a purely discriminative model, into the model of conditional random fields (CRFs). It has been shown that the recently proposed semi-supervised discriminative criterion [26] is efficient under model misspecification and covariate shift scenarios for "simple" (i.e. without underlying structure) output tasks. We apply the semi-supervised approach to the criterion of conditional random fields and carry out experiments on structured output problems. We discuss the limits and drawbacks of the criterion.

We propose to integrate the marginal probability of observations into an active learning framework for the structured output prediction. We demonstrate on the standard natural language processing data sets that the proposed pool-based active learning approach based on quota sampling is efficient.

The paper is organized as follows: in Section 2 we consider the asymptotically optimal semi-supervised criterion [26], Section 3 introduces the model of conditional random fields [11], widely used for structured output prediction. In Section 4 we discuss the application of the semi-supervised criterion to the model of conditional random fields. Section 5 discusses the limits of the semi-supervised discriminative criterion applied to the CRFs and introduces our approach of pool-based active learning based on quota sampling. Section 6 illustrates our experiments on synthetic as well as real-world applications. We discuss the state-of-the art approaches and related work in Section 7. Concluding remarks and perspectives close the paper.

2 Semi-supervised Discriminant Estimator

To start with, let us place in a context of classification without taking any structure into consideration. Let the observation variable X take its values in a finite set $\mathcal{X}$; Y is the class variable which takes its values in $\mathcal{Y}$. We suppose $\mathcal{Y}$ to be a finite set, and $\{X_i, Y_i\}_{i=1}^n$ are observations and their labels available for the training.

Let $g(y|x;\theta)$ be the conditional probability function, parameterized by θ. Then the standard conditional maximum likelihood estimator is defined by

$$\hat{\theta}_n = \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i|X_i;\theta), \tag{1}$$

where $\ell(y|x;\theta) = -\log g(y|x;\theta)$ denotes the negated conditional log-likelihood function.

The asymptotically optimal semi-supervised estimator $\hat{\theta}_n^s$ proposed by [26] is defined by

$$\hat{\theta}_n^s = \arg\min_{\theta\in\Theta} \sum_{i=1}^{n} \frac{q(X_i)}{\sum_{j=1}^{n} \mathbb{1}\{X_j = X_i\}} \ell(Y_i|X_i;\theta), \tag{2}$$

where $q(x)$ is the marginal probability of observations and can be considered as some prior knowledge. We suppose that infinitely many observations are available, and that the true value $q(x)$ can be estimated. The semi-supervised estimator presented as eq. (2) is a weighted version of the usual conditional maximum likelihood estimator.

The semi-supervised estimator is shown to be asymptotically optimal and to be particularly efficient for the misspecified cases, that is if $g(y|x;\theta_\star) \neq \eta(y|x)$, where $\eta(y|x)$ is the true conditional probability that generated the data; in the following, $\pi(y,x) = \eta(y|x)q(x)$.

To be precise, the essential properties of the standard and weighted (semi-supervised) estimators consist in the following:

$$\sqrt{n}\left(\hat{\theta}_n - \theta_\star\right) \xrightarrow{L} \mathcal{N}\left(0, J^{-1}(\theta_\star)I(\theta_\star)J^{-1}(\theta_\star)\right), \tag{3}$$

$$\sqrt{n}\left(\hat{\theta}_n^s - \theta_\star\right) \xrightarrow{L} \mathcal{N}\left(0, J^{-1}(\theta_\star)H(\theta_\star)J^{-1}(\theta_\star)\right), \tag{4}$$

where

$$H(\theta_\star) = \mathrm{E}_q\left(\mathrm{V}_\eta\left[\nabla_\theta \ell(Y|X;\theta_\star)|X\right]\right), \tag{5}$$

$$I(\theta_\star) = \mathrm{E}_\pi\left[\nabla_\theta \ell(Y|X;\theta_\star)\left\{\nabla_\theta \ell(Y|X;\theta_\star)\right\}^{\mathrm{T}}\right], \tag{6}$$

$$J(\theta_\star) = \mathrm{E}_\pi\left[\nabla_{\theta^{\mathrm{T}}}\nabla_\theta \ell(Y|X;\theta_\star)\right]. \tag{7}$$

The case of a covariate shift (observation variables are sometimes called explanatory variables or covariates) is a rather frequent situation in real-world applications. The covariate shift arises if $q_0(x) \neq q_1(x)$, where $q_0(x)$ is determined by the sampling scheme and $q_1(x)$ is determined by the population.

In the absence of covariate shift:

$$\lim_{n\to\infty} \frac{q_1(x_i)}{n^{-1}\sum_{j=1}^{n} \mathbb{1}\{x_j = x_i\}} \longrightarrow 1. \tag{8}$$

With a covariate shift, we have:

$$\lim_{n\to\infty} \frac{q_1(x_i)}{n^{-1}\sum_{j=1}^{n} \mathbb{1}\{x_j = x_i\}} \longrightarrow \frac{q_1(x_i)}{q_0(x_i)}. \tag{9}$$

The weighting scheme by the importance ratio is considered in [24].

The semi-supervised estimator of eq. (2) is shown to be asymptotically optimal under the covariate shift [25]. The advantage of the semi-supervised approach can be observed only when considering the scaled excess logarithmic risk

$$n(\mathrm{E}_\pi[\ell(Y|X;\hat{\theta}_n)] - \mathrm{E}_\pi[\ell(Y|X;\theta_\star)]) \tag{10}$$

or the scaled squared error

$$n\|\hat{\theta}_n - \theta_\star\|^2. \tag{11}$$

The true marginal probability of observations have to be provided to compute both the scaled excess logarithmic risk and the scaled squared error. However, as we have already mentioned, this is not possible for real-world applications.

3 Conditional Random Fields

Conditional random fields (CRF) [11,27] are a discriminative model based on the following probabilistic distribution

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\theta(\mathbf{x})} \exp\left\{ \sum_{t=1}^{T}\sum_{k=1}^{K} \theta_k f_k(y_{t-1}, y_t, x_t) \right\}, \tag{12}$$

where $\mathbf{x} = (x_1, \ldots, x_T)$ denotes the sequence of observations (input) and $\mathbf{y} = (y_1, \ldots, y_T)$ is the sequence of labels (output); $\{f_k\}_{1\leq k\leq K}$ is an arbitrary set of feature functions and $\{\theta_k\}_{1\leq k\leq K}$ are the associated real-valued parameter values. By convention, y_0 refers to a particular (always observed) label which indicates the beginning of the sequence.

The CRF form presented as eq. (12) is usually referred to as linear-chain CRF, although y_t and x_t could be composed not only of the individual sequence tokens, but of sub-sequences (n-grams) of some fixed length or other localized characteristics.

We will denote by $\mathcal{Y}$, $\mathcal{X}$, respectively, the sets in which y_t and x_t take their values. The normalization factor in eq. (12) is defined by

$$Z_\theta(\mathbf{x}) = \sum_{\mathbf{y}\in Y^{\mathrm{T}}} \exp\left\{\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k f_k(y_{t-1}, y_t, x_t)\right\}. \tag{13}$$

One of the possible ways to define features is the combination of bigram $\lambda_{y',y,x}$ and unigram $\mu_{y,x}$ features

$$\begin{aligned}\sum_{k=1}^{K}\theta_k f_k(y_{t-1}, y_t, x_t) = &\sum_{y\in Y, x\in X}\mu_{y,x}\mathbb{1}\{y_t = y, x_t = x\}+ \\ &\sum_{y',y\in Y^2, x\in X}\lambda_{y',y,x}\mathbb{1}\{y_{y-1} = y', y_t = y, x_t = x\},\end{aligned} \tag{14}$$

where $\mathbb{1}(\text{test}) = 1$, if the variables are observed jointly and 0 otherwise. We can rewrite equation (14) as $\mu_{y_t,x_t} + \lambda_{y_{t-1},y_t,x_t}$, and we use this more compact representation in the following. We use this feature combination, unigram and bigram templates, in our experiments in Section 6.

Given N independent labeled sequences $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, the conditional maximum likelihood estimation is based on the minimization, with respect to θ, of the negated log-likelihood

$$\begin{aligned}\ell(\mathcal{D};\theta) &= -\sum_{i=1}^{N}\log p_\theta(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \\ &= \sum_{i=1}^{N}\left\{\log Z_\theta(\mathbf{x}^{(i)}) - \sum_{t=1}^{T_i}\sum_{k=1}^{K}\theta_k f_k(y_{t-1}^{(i)}, y_t^{(i)}, x_t^{(i)})\right\},\end{aligned} \tag{15}$$

where T_i is the length of an observation $\mathbf{x}^{(i)}$.

Although $\ell(\mathcal{D};\theta)$ is a smooth convex function, it has to be optimized numerically, and standard gradient-based methods, such as a quasi-Newton approach, can be applied directly.

The gradient of $\ell(\mathcal{D};\theta)$ is given by

$$\begin{aligned}\frac{\partial\ell(\theta)}{\partial\theta_k} = &\sum_{i=1}^{N}\sum_{t=1}^{T_i}\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}\, f_k(y_{t-1}, y_t, x_t^{(i)}) - \\ &\sum_{i=1}^{N}\sum_{t=1}^{T_i} f_k(y_{t-1}^{(i)}, y_t^{(i)}, x_t^{(i)}),\end{aligned} \tag{16}$$

where $\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}\, f_k(y_{t-1}, y_t, x_t^{(i)})$ denotes the conditional expectation.

In our experiments, the log-likelihood is penalized by the L_2 norm to avoid overfitting.

4 Semi-supervised Conditional Random Fields

The semi-supervised criterion presented as eq. (2) applied to the conditional random fields criterion, referred later to as weighted CRFs, takes the form:

$$C(\theta) = \sum_{\mathbf{x} \in \mathcal{X}} -q(\mathbf{x}) \frac{1}{N_{\mathbf{x}}} \log p_\theta(\mathbf{y}|\mathbf{x}), \tag{17}$$

where $p_\theta(\mathbf{y}|\mathbf{x})$ is defined by eq. (12), and $N_{\mathbf{x}}$ is the number of times a sequence $\mathbf{x}$ has been observed in the training corpus.

The marginal probability of observations $q(\mathbf{x})$ has to be provided or approximated and introduced into the model of conditional random fields. In our case, the observations are sequences, what makes the task even more difficult.

If our data are artificial, generated by a hidden Markov model of the first order, then estimation of the probability of the observation sequences is straightforward. Following the standard notations [20], let A be the state transition probabilities, B be the observation probability matrix, $p(y)$ be the initial state distribution, $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ be an observation sequence of the length T. The probability of a series of observations, i.e., of a sequence is given by

$$\begin{aligned} q(\mathbf{x}) &= \sum_Y p(\mathbf{x}, \mathbf{y}) \\ &= \sum_Y p(y_1) b_{y_1}(x_1) a_{x_1,x_2} b_{y_2}(x_2) \ldots a_{x_{T-1},x_T} b_{y_T}(x_T). \end{aligned} \tag{18}$$

Usually in real-world problems, the structure of observations is unknown. It is not possible to compute the marginal probability of observations exactly, and it has to be estimated empirically.

Note, that $N_{\mathbf{x}}$ equals 1 in a number of real-world applications, since each sequence is observed usually only once in a training corpus.

5 Motivation for Pool-Based Active Learning

The application of the semi-supervised discriminative estimator to real-world data sets does not always ameliorate the performance.

There are several reasons, why the performance of the criterion does not dominate the performance of the standard approach. The semi-supervised criterion performs better in the case of a misspecified model (the more a model is misspecified, the more efficient is the semi-supervised criterion compared to the standard, not weighted approach) and under a covariate shift. Usually, both scenarios are typical for a real data set. However, carrying out experiments on simulated data, we have noticed that the advantage of the semi-supervised approach is observed only when considering the scaled excess logarithmic risk, eq. (10), and the squared error, eq. (11). To compute these values, the true distribution of the observations has to be provided. In any real-world task the distribution is not available. From a number of experiments on the real data we made a conclusion that although the marginal probability of observations can be efficiently approximated, the approximation is still not good enough to be used in the semi-supervised estimator instead of the true one.

However, we guess that even an approximation of the marginal distribution can be informative. We propose to use the probability of observations to sample a pre-defined pool of training instances (of a small size n) to achieve the best possible generalization error. Our idea is close, in some sense, to [32], who considered an active learning approach based on self-training.

In the discriminative semi-supervised criterion presented as eq. (2), training instances with high probability can be automatically considered to be more important than those with low probability. In this sense, the discriminative semi-supervised approach is associated with stratified sampling. However, one of natural language phenomena consists in that rare events are as important as frequent events, and can therefore not be neglected.

We propose to apply the non-probabilistic quota sampling to select training sequences efficiently. In the method we propose, candidates for training instances are sorted according to their marginal probabilities and are divided into n groups, where n is the number of observations we use for the training procedure. We choose (randomly) one training instance from each group. Under quota sampling we mean here that we sample (uniformly) data instances from each frequency group. Therefore, we guarantee that we train our model taking frequent as well as rare dependencies into consideration.

We illustrate on standard natural language processing problems, in Sections 6.3 and 6.4 that the quota sampling (QS) pool-based active learning approach outperforms training procedures which choose their training instances randomly. In Section 6.2 we show that the proposed method outperforms a state-of-the art approach FuSAL.

6 Experiments on Artificial and Real Data Sets

In this section, we describe our experiments and provide our results on synthetic and two standard natural language processing problems, namely on NetTalk Phonetisation Task and on CoNLL 2003 challenge.

Section 6.1 illustrates limits of the semi-supervised criterion, even for an artificial data set. We demonstrate on real data that the proposed QS pool-based active learning is an efficient approach (Sections 6.3 and 6.4), in particular in case where the number of observations n is small.

The state-of-the art performance (mostly in the context of fully supervised learning) of the considered real data sets can been found, for instance, in [25].

We would like to underline that we are especially interested in cases where n, the number of observed instances, is small.

6.1 Weighted Conditional Random Fields Experiments

The synthetic sequential data are simulated with hidden Markov models of the first order. The observation alphabet contains 5 symbols, the size of the labels alphabet is 6. All simulated sequences are of the same length which equals 5. The minimal achievable error is about 6%. The value of Bayes error is approximated by a percentage of errors obtained by decoding using the true values of the state transition and observation probability matrices.

Since we know the distribution which generates the data and the true parameters are available, we use the forward algorithm and eq. (18) to compute the marginal probability of observations $q(\mathbf{x})$.

The results of our experiments with the synthetic sequential data are illustrated by Figure 1. The size of the training corpus varies from 10 to 200 training instances. The percentage of error is always estimated on test data (test data contains 10000 instances). The number of Monte-Carlo replications in the experiment is 150. The boxplotted difference, which is shown on Figure 1, is positive, if the weighted CRFs performs better, i.e. has a lower error rate than the standard approach.

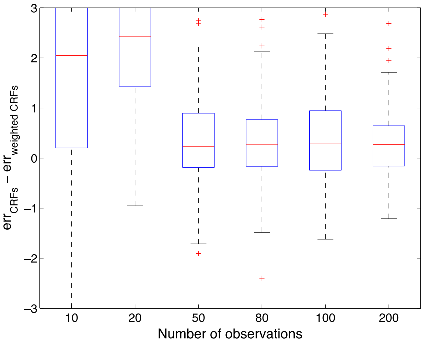


Fig. 1. Simulated data. Difference of error rates of standard and weighted conditional random fields by marginal probability.

The difference in performance of the standard and the semi-supervised CRFs is significant only for $n = 10, 20$, and not significant for larger n, even in the ideal situation, where we know the exact marginal probability of observations.

As to the real-world data experiments, where we dispose only the approximated values of the probability of observations, we consider the difference in performance to be not significant.

6.2 Fully Supervised Active Learning Approach (FuSAL)

We compare the performance of the proposed pool-based active learning to the one of a state-of-the art method called FuSAL (Fully Supervised Active Learning) introduced in [32]. Algorithm 1 describes the approach. A utility function we use in our experiments is the same as in [32]

$$u_\theta(\mathbf{x}) = 1 - p_\theta(\hat{\mathbf{y}}|\mathbf{x}), \tag{19}$$

where $\hat{\mathbf{y}}$ is computed using the decoding Viterbi algorithm, and θ corresponds to the current model. The intuition behind the utility function is to consider sequences for which the current model is least confident to be more important than other observations.

Algorithm 1. General Active Learning Framework

```
$m$ – number of examples selected within one loop
$\mathcal{D}_l$ – set of labeled instances
$\mathcal{D}_u$ – set of unlabeled instances
$u_\theta(\mathbf{x})$ – utility function

while stopping criterion is not met do
    train model $M$ using $D_l$
    estimate $u_\theta(\mathbf{x}_i)$ $\forall \mathbf{x}_i \in \mathcal{D}_u$
    choose $m$ examples whose $u_\theta(\mathbf{x})$ is maximal
    get labels for the $m$ chosen instances
    move the $m$ labeled examples from $\mathcal{D}_u$ to $\mathcal{D}_l$
end while
```

In our experiments, we add instances which are to be labeled one by one ($m = 1$). The first instance is chosen randomly from the training corpus. The stopping criterion is the number n of observed sequences. If the cardinality of $\mathcal{D}_l$ is equal to n, the stopping criterion is met.

6.3 Active Learning Experiments on Nettalk Phonetisation Task

The original Nettalk corpus has been introduced in [23]. The Nettalk corpus we use in our experiments has been suggested for the Pascal Letter-to-Phoneme Conversion Challenge[1]. The English data set contains 16280 words aligned with their phonetical transcriptions. The alphabet of observation symbols includes 26 letters, and the number of phonemes, i.e., the number of labels, is 53 including the alignment symbol. The corpus is split into 10 parts. Each part includes 1628 sequences of observations and corresponding labels. One part, i.e., 1628 sequences, is used to test the performance. We use all available observation sequences of the corpus to estimate empirically the probability of observations $q(\mathbf{x})$.

To approximate $q(\mathbf{x})$, we follow the idea of n-gram linguistic models [7]. We let $q(\mathbf{x}) = q(x_1, \ldots, x_T) = \prod_t p(x_t|x_{t-1}, x_{t-2}, x_{t-3})$, where

$$\begin{aligned} p(x_t|x_{t-1}, x_{t-2}, x_{t-3}) \approx \\ C(x_t, x_{t-1}, x_{t-2}, x_{t-3})/C(x_{t-1}, x_{t-2}, x_{t-3}), \end{aligned} \tag{20}$$

$C(\cdot)$ means counts estimated on all available observations.

The estimated $q(\mathbf{x})$ are sorted into n frequency groups, and we sample one training instance from each frequency group. The training is performed with two

[1] http://pascallin.ecs.soton.ac.uk/Challenges/PRONALSYL/

types of features, bigram and unigram, as shown by eq. (14). The regularization parameter ρ of the penalty term $\rho||\theta||^2$ is the same for all tested approaches, QS active learning, random sampling, and FuSAL, and is fixed to $\rho = 0.1$ (the value is chosen by cross validation).

Figure 2 illustrates the performance of the FuSAL method on the Nettalk data sets (50 Monte-Carlo replications). One of its obvious disadvantages and hence causes of its poor performance is that the method is not suitable for cases where n is small. To train the initial model, the method requires a number of labeled sequences, and if these sequences are not selected carefully, the training results in a model whose error rate on the testing set is large. It is not reasonable to compute the utility function, and therefore perform active learning based on a model which is not efficient.

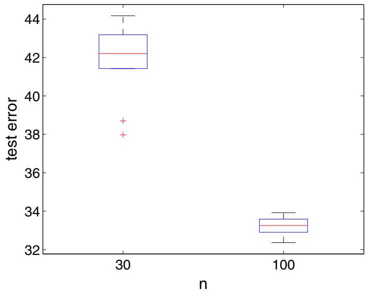


Fig. 2. FuSAL performance (error rates). Nettalk corpus, $n = 30, 100$.

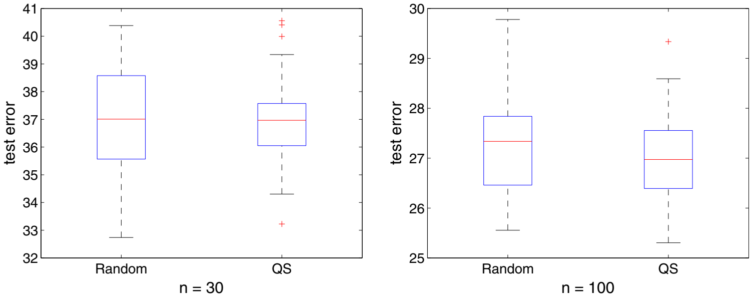


Fig. 3. Nettalk corpus. Comparison of error rates for $n = 30$ and $n = 100$. The pool-based active learning based on quota sampling (QS) is more efficient than random choice of training sequences.

Figure 3 illustrates our results of the pool-based active learning approach compared to random sampling. We performed 50 Monte-Carlo replications. For a small number of observations, $n = 30$ and $n = 100$, we noticed that the test error and its variance are smaller if observations are chosen according to the proposed pool-based active learning method and not randomly.

It is easy to see that the QS approach approach outperforms the random sampling and FuSAL.

For the qualitative analysis of sequences selected by the proposed quota sampling method and the standard approach presented as Algorithm 1, see Tables 1 and 2 respectively. Note that applying the utility function, eq. (19), we tend to select sequences of similar morphological structure.

Table 1. Nettalk corpus. Sequences chosen by QS, $n = 30$.

ail	inconceivably	neat	superlative	chase
sworn	interstate	strain	unnaturally	fresco
secret	invertebrate	comrade	ennoble	haughtily
dribble	meditate	parasite	woodwork	meteoric
shoemaker	unstained	simpleton	soberly	snake
chloroform	aspire	babe	cheese	rise

Table 2. Nettalk corpus. Sequences chosen by FuSAL (Algotihm 1), $n = 30$, $m = 1$.

hogshead	shepherdess	aggressiveness
misrepresentation	representation	representative
misapprehension	interdependence	superintendence
superintendent	misunderstanding	experimentation
standardization	interpretation	transcontinental
undenominational	unconstitutional	counterrevolution
indiscriminately	characteristically	internationally
characterization	instantaneously	enthusiastically
constitutional	conscientiously	incomprehensible
intermittently	instrumentality	correspondingly

6.4 Active Learning Experiments on CoNLL 2003 Corpus

Named entity recognition consists in extracting groups of syntagmas that correspond to named entities (e.g., names of persons, organizations, places, etc.). The data used for our experiments are taken from the CoNLL 2003 challenge [31] and imply four distinct types of named entities. Labels have the form B-X or I-X, where B means "begin" and I means "inside" of a named entity X (note that the label B-PER is not present in the corpus). Words that are not included in any named entity, are labeled with O (outside). Overall, there are 8 labels.

At each position in the text, the input consists of three separate components, so we have three types of observations: a word (with 30290 distinct words in the corpus), its part-of-speech (44), and syntactic (18) tags. The training set includes about 15000 sequences (phrases). Development set (Test A) and test set (Test B) include about 3500 sequences each.

We use all available sequences to estimate $q(\mathbf{x})$. We apply the same approach as for the Nettalk corpus, described in the previous section. However, for the

CoNLL 2003 data set we use the Markovian dependency of the second, and not of the third, order. Since the data set has three types of observations, we have to take into consideration marginal probabilities of each type of observation. The probability $q(\mathbf{x})$ is approximated by the product of marginal probabilities of its components $p(\mathbf{x}_{\text{word}})p(\mathbf{x}_{\text{POS tag}})p(\mathbf{x}_{\text{synt. tag}})$.

The training is carried out with two dependencies, unigram and bigram, extracted for each type of observation, i.e. our feature choice is as follows:

$$\begin{aligned}&\mu_{y_t,x_{\text{word},t}} + \mu_{y_t,x_{\text{POS tag},t}} + \mu_{y_t,x_{\text{synt. tag},t}}\\ &+\lambda_{y_t,x_{\text{word},t}} + \lambda_{y_t,x_{\text{POS tag},t}} + \lambda_{y_t,x_{\text{synt. tag},t}}.\end{aligned} \tag{21}$$

The regularization parameter ρ of the penalty term $\rho\|\theta\|^2$ is chosen by cross validation and is the same for all tested sampling methods, $\rho = 0.5$.

Figure 4 illustrates the results of our experiments with FuSAL. Figure 5 shows the results for random sampling and QS. We carried out 50 Monte-Carlo replications for all methods. For a small number of observations, $n = 10$ and $n = 50$, as illustrated on the figure, it is obvious that the error rate on the test data

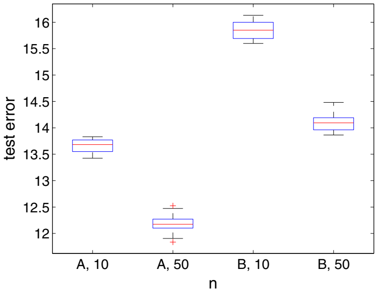


Fig. 4. FuSAL performance (error rates). CoNLL 2003, for test A and test B sets, $n = 10, 50$.

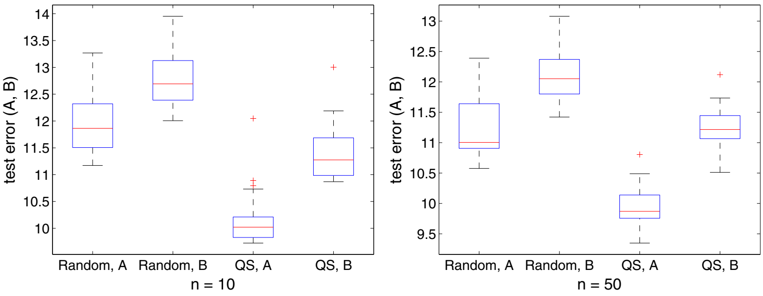


Fig. 5. CoNLL 2003 data set. Comparison of error rates (for test A and test B sets) while training on $n = 10$ and $n = 50$ sequences. Active learning based on marginal probability (QS on the boxplots) is much more efficient than arbitrary choice of observations for training.

(both test A and test B sets) is much smaller while using the quota sampling active learning than choosing training instances arbitrary. FuSAL is less efficient as well.

Our experiments show that FuSAL is an acceptable active learning method if some initial, not very small, $\mathcal{D}_l$ is provided and if a reasonable initial model M can be created.

7 Related Work

Many proposals of semi-supervised methods have been recently made to sequence labeling. As to active learning for structured output prediction, there are much less published ideas.

A maximum margin semi-supervised learning approaches for structured output prediction are described in [1] and [3]; [10], [15], and [13] discuss semi-supervised learning for conditional random fields.

The minimum entropy regularization approach of Grandvalet and Bengio [8] has been applied to conditional random fields by [10] :

$$-\sum_{i=1}^{|\mathcal{D}_l|} \log p_\theta(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) + \frac{||\theta||^2}{2\sigma^2} - \\ \rho \sum_{i=|\mathcal{D}_l|+1}^{|\mathcal{D}_l|+|\mathcal{D}_u|} \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) \log p_\theta(\mathbf{y}|\mathbf{x}^{(i)}), \tag{22}$$

where $\mathcal{D}_l$ are labeled data and $\mathcal{D}_u$ are unlabeled instances; σ^2 and ρ are parameters fixed usually by cross validation. The direct computation of the gradient of the entropy term of the criterion requires $O(T^2|Y|^3)$ operations in comparison to $O(T|Y|^2)$ of a standard forward-backward procedure. [13] proposed an efficient way (complexity of a standard forward-backward algorithm) to compute the gradient of the criterion presented in (22).

A hybrid semi-supervised model is proposed in [28]. The model combines discriminative and generative models, the parameters $\Gamma = \{\{\gamma_i\}_{i=1}^{I}, \{\gamma_j\}_{j=I+1}^{I+J}\}$ are associated with I generative and J discriminative models. Unlabeled data are introduced into the generative models. The following criterion

$$p(\mathbf{y}|\mathbf{x}, \Lambda, \Theta, \Gamma) \propto \prod_i p_i^D(\mathbf{y}|\mathbf{x}, \lambda_i)^{\gamma_i} \prod_j p_j^G(\mathbf{x}, \mathbf{y}, \theta_j)^{\gamma_j} \tag{23}$$

contains three sets of parameters to be estimated, Γ, Λ, and Θ. The values of Λ are estimated on labeled data. An iterative optimization procedure runs until convergence is used to adjust Γ (parameters of hybrid models) and parameters Θ associated with discriminative components.

Recently [29] introduced a semi-supervised approach that is simpler than the one proposed in [28], since there are only two parameter vectors to be estimated. The parameter vector Λ is estimated on labeled data using a discriminative model, and Θ on unlabeled data, using a generative approach. However, the number of parameters to be estimated is quite large.

The approach discussed in [28] was called a great step forward in hybrid models [5], since it combines models that take the underlying structure into account, namely hidden Markov models and conditional random fields. The approach of [29] has been recently applied to parsing problems by [30].

One of the recent works on semi-supervised learning applied to natural language processing is a trial to add incomplete annotations [33]. Ambiguous annotations are considered as candidate labels, and parameters are estimated by marginalizing out the unknown labels. The method is a particular case of hidden conditional random fields, introduced in [19].

The idea to introduce the knowledge of labels proportions, the method called "expectation regularization", proposed in [14] for maximum entropy models, has been generalized in [15] for structured output prediction, using linear-chain CRFs. The approach was called generalized expectation. It was supposed that not only fully labeled instances can be used but labeled features as well.

Recently [32] proposed an approach that combines semi-supervised and active learning. The semi-supervised active learning method [32] which is actually self-training active learning approach, selects instances of high utility to be labeled and to be used for training. Estimation of utility of a given sequence is a problem in itself, since it can be done in many different ways.

It is discussed in [32] whether it is more reasonable to label manually only subsequences (e.g., features) of high utility instead of labeling entire sequences. A similar idea, an efficient learning of features from unlabeled data is considered in [18].

8 Conclusion

In this contribution, we addressed two problems, semi-supervised learning and active learning in discriminative models, more specifically, in conditional random fields. We demonstrated on the artificial data set that the considered discriminative semi-supervised method can be applied to conditional random fields. However, its application to real tasks is still an open problem, since an efficient approximation of the probability of observations, whose structure is complex and unknown, is still a challenge.

The proposed pool-based active learning method is based on the intuition that rare observations are not less important than frequent observations. In particularly, this is the case in the domain of natural language processing. We have shown that selecting training instances using quota sampling is much more efficient in terms of error rates on test data than choosing them randomly. The proposed approach is also more efficient than FuSAL, a state-of-the art method. Most of state-of-the art active learning methods (e.g., FuSAL) are based on the idea that there already exists a set of labeled instances, and are therefore not suitable for cases where the number of labeled points is very limited.

An important advantage of the proposed quota sampling approach is simplicity of implementation. The open issue is the theoretical analysis of the proposed quota sampling pool-based active learning approach, which is quite efficient on the real-world data sets.

References

1. Altun, Y., McAllester, D., Belkin, M.: Maximum margin semi-supervised learning for structured variables. In: NIPS (2005)
2. Bouchard, G., Triggs, B.: The trade-off between generative and discriminative classifiers. In: IASC (2004)
3. Brefeld, U., Scheffer, T.: Semi-supervised learning for structured output variables. In: ICML (2006)
4. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
5. Daumé III, H.: Semi-supervised or semi-unsupervised? In: NAACL Workshop on Semi-supervised Learning for NLP (2009)
6. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of The Royal Statistical Society Series B 39(1), 1–38 (1977)
7. Goodman, J.T.: A bit of progress in language modeling. Technical Report MSR-TR-2001-72, Microsoft Research, Redmond (August 2001)
8. Grandvalet, Y., Bengio, Y.: Semi-supervised learning by entropy minimization. In: NIPS (2004)
9. Holub, A., Perona, P.: A discriminative framework for modelling object classes. In: CVPR 2005 (2005)
10. Jiao, F., Wang, S., Lee, C.H., Greiner, R., Schuurmans, D.: Semi-supervised conditional random fields for improved sequence segmentation and labeling. In: ACL/COLING (2006)
11. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: ICML (2001)
12. Lasserre, J.A., Bishop, C.M., Minka, T.P.: Principled hybrids of generative and discriminative models. In: CVPR (2006)
13. Mann, G., McCallum, A.: Efficient computation of entropy gradient for semi-supervised conditional random fields. In: NAACL/HLT (2007)
14. Mann, G., McCallum, A.: Simple, robust, scalable semi-supervised learning via expectation regularization. In: ICML (2007)
15. Mann, G., McCallum, A.: Generalized expectation criteria for semi-supervised learning of conditional random fields. In: ACL (2008)
16. Minka, T.: Discriminative models, not discriminative training. Technical Report TR-2005-144, Microsoft Cambridge (2005)
17. Ng, A., Jordan, M.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In: NIPS (2002)
18. Qi, Y., Kuksa, P.P., Collobert, R., Kavukcuoglu, K., Weston, J.: Semi-supervised sequence labelling with self-learned feature. In: ICDM (2009)
19. Quattoni, A., Collins, M., Darrell, T.: Conditional random fields for object recognition. In: NIPS (2004)
20. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
21. Scudder, H.J.: Probability of error of some adaptive pattern-recognition machines. IEEE Transactions on Information Theory 11, 363–371 (1965)

22. Seeger, M.: Learning with labeled and unlabeled data. Technical report, University of Edinburgh, Institute for Adaptive and Neural Computation (2002)
23. Sejnowski, T.J., Rosenberg, C.R.: Parallel networks that learn to pronounce english text. Complex Systems 1 (1987)
24. Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of Statistical Planning and Inference 90, 227–244 (2000)
25. Sokolovska, N.: Contributions to estimation of probabilistic discriminative models: semi-supervised learning and feature selection. PhD thesis, TELECOM ParisTech (2010)
26. Sokolovska, N., Cappé, O., Yvon, F.: The asymptotics of semi-supervised learning in discriminative probabilistic models. In: ICML (2008)
27. Sutton, C., McCallum, A.: An introduction to conditional random fields for relational learning. In: Getoor, L., Taskar, B. (eds.) Introduction to Statistical Relational Learning. The MIT Press, Cambridge (2006)
28. Suzuki, J., Fujino, A., Isozaki, H.: Semi-supervised structured output learning based on a hybrid generative and discriminative approach. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (2007)
29. Suzuki, J., Isozaki, H.: Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In: ACL (2008)
30. Suzuki, J., Isozaki, H., Carreras, X., Collins, M.: An empirical study of semi-supervised structured conditional models for dependency parsing. In: EMNLP (2009)
31. Tjong Kim Sang, E.F., de Meulder, F.: Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: CoNLL (2003)
32. Tomanek, K., Hahn, U.: Semi-supervised active learning for sequence labeling. In: ACL and AFNLP (2009)
33. Tsuboi, Y., Kashima, H., Mori, S., Oda, H., Matsumoto, Y.: Training conditional random fields using incomplete annotations. In: COLING (2008)

Comparing Probabilistic Models for Melodic Sequences

Athina Spiliopoulou and Amos Storkey

School of Informatics, University of Edinburgh, United Kingdom
{a.spiliopoulou,a.storkey}@ed.ac.uk

Abstract. Modelling the real world complexity of music is a challenge for machine learning. We address the task of modeling melodic sequences from the same music genre. We perform a comparative analysis of two probabilistic models; a Dirichlet Variable Length Markov Model (Dirichlet-VMM) and a Time Convolutional Restricted Boltzmann Machine (TC-RBM). We show that the TC-RBM learns descriptive music features, such as underlying chords and typical melody transitions and dynamics. We assess the models for future prediction and compare their performance to a VMM, which is the current state of the art in melody generation. We show that both models perform significantly better than the VMM, with the Dirichlet-VMM marginally outperforming the TC-RBM. Finally, we evaluate the short order statistics of the models, using the Kullback-Leibler divergence between test sequences and model samples, and show that our proposed methods match the statistics of the music genre significantly better than the VMM.

Keywords: melody modeling, music feature extraction, time convolutional restricted Boltzmann machine, variable length Markov model, Dirichlet prior.

1 Introduction

In this paper we are interested in learning a generative model for melody directly from musical sequences. This task is challenging for machine learning methods. Repetition of musical phrases, which is essential for Western music, can occur in almost arbitrary points in time and with different degrees of variation. Furthermore, although pieces from the same genre are built using the same structural principles, the statistical relations among and within melodies from different pieces are highly complex, as melody depends on several different components, such as scale, rythm and meter, which in many cases interdepend on each other.

Capturing the statistical regularities within a musical genre is a first step towards realistic music generation. Additionally, identifying and representing these dependencies in an unsupervised manner is particularly desirable, as descriptive features of the underlying structure of music can not only help in the analysis and synthesis of music, but also enhance the performance on a variety of musical tasks such as genre classification and music retrieval.

In this work we consider two methods for the problem of melody modeling; a Time Convolutional Restricted Boltzmann Machine (TC-RBM) and a Dirichlet Variable Length Markov Model (Dirichlet-VMM). The first is an adaptation of the Convolutional RBM (Lee et al., 2009) for modeling sequential data and is motivated by the

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 289–304, 2011.

ability of RBM type models to extract high quality latent features from the input space. The second one is a non-latent variable model and is a novel form of VMM, the latter one being regarded as state of the art in melody generation (Paiement, 2008).

Our purpose is to answer the following questions. Are these probabilistic models able to learn the inherent structure in melodic sequences and generate samples that respect the statistics of the music genre? What aspects of the musical stucture can each of the models learn? Can melodies be decomposed into a set of musical features in the same way that images can be decomposed into sets of edges and documents into sets of topics?

We train the models on a set of traditional reel tunes and perform a comparative analysis of these with a standard VMM. We show that the TC-RBM learns descriptive music features, such as underlying chordal structure, musical motifs and transformations of those. We assess the models on future prediction and find that our proposed methods perform significantly better than the standard VMM and are comparable to each other, with the Dirichlet-VMM having slightly higher log-likelihood. Likewise, we evaluate the short order statistics of model samples, using the Kullback-Leibler divergence, and show that samples from the TC-RBM and the Dirichlet-VMM match the statistics of the test data significantly better than samples from the VMM.

2 Related Work

In many cases, the difficulties associated with modeling music have been dealt with by incorporating domain knowledge in the models. In this line of research, Paiement (2008) proposes modeling different aspects of music, such as chord progressions, rhythm and melody, using graphical models and Input-Output HMMs. The structure of the models and the data representations used are based on musical theory. Additionally, Weiland et al. (2005) propose a Hierarchical Hidden Markov Model (HHMM) for pitch. The HHMM is structurally simple and its internal states are pre-defined with respect to music assumptions.

A different course of research examines more general machine learning methods, which are able to automatically capture complex relations in sequential data, without introducing much prior knowledge. In this paper we are taking this approach and consider models that do not make assumptions explicit to music.

Lavrenko and Pickens (2003) propose Markov Random Fields (MRFs) for modeling polyphonic music. In order for the MRF to remain tractable, much information needs to be discarded, thus making the model less suitable for realistic music.

Eck and Schmidhuber (2002) show that a Long-Short Term Memory (LSTM) Recurrent Neural Network can successfully model long-term structure in two simple musical tasks. In Eck and Lapalme (2008) the LSTM is extended to include meter information. The output of the network is conditioned on the current chord and specific previous time-steps, chosen according to the metrical boundaries. Trained on a set of traditional Irish reels the LSTM is shown to generate pieces that respect the reel style.

Finally, Dubnov et al. (2003) propose Incremental Parsing (IP) and Prediction Suffix Trees (PSTs) for modeling melodies, the latter one being the data structure used to represent VMMs. Both algorithms train simple dictionary-based predictors that parse music into a lexicon of phrases or motifs. Paiement (2008) argues that despite their simple nature, these two models generate impressive musical results when sampled and can be considered state of the art in melody generation.

3 Preliminaries

3.1 Musical Motifs

Before describing the models, we explain the concept of motifs and their importance to music modeling, as we believe it is useful in understanding the types of structures that the VMM and the TC-RBM are trying to capture.

In Western Music, the smallest building block of a piece is called a motif. Motifs typically comprise three, four or more notes and most pieces can be expressed as a combination of different motifs and their transformations. Frequent transformations include replacement, splitting and merging of notes, and typically respect the metrical boundaries of a piece. We believe that successful capturing of music motifs can be very useful when modeling melodies, as specific motifs and their transformations are highly likely to be repeated within a piece, as well as among pieces from the same musical form.

3.2 Variable Length Markov Model

The VMM (Ron et al., 1994) is a statistical model for discrete sequential data and has been shown to generate state of the art musical results when modeling melodies (Dubnov et al., 2003). Its advantage to a standard Markov Model (*n-gram*) is that the order of the former is not fixed, but instead depends on the observed context.

A VMM is represented by a Prediction Suffix Tree. The edges of the tree are labeled with symbols from the alphabet, in this case the different music notes. Each node defines the conditional probability distribution of the next symbol given the context we acquire by concatenating all the edge symbols from the root to the node [1]. The tree has depth L, but is not complete[2], thus giving rise to contexts that are shorter than L but are still used for prediction.

To learn the tree, we start from a single root node labeled by the empty string and 'grow' the tree using a breadth-first search for contexts that satisfy the following criteria:

- The length of a context is upper bounded by a fixed length L
- The frequency counts of a context exceed a fixed threshold c_{min}
- The ratio of the conditional probability distribution defined at a node with that defined at its parent node exceeds a fixed threshold ϵ_{min}

The resulting tree comprises contexts corresponding to musical phrases that appear frequently in the data and convey significant information about the value of future

[1] Note that during prediction only the conditional probability distributions defined at the leaf nodes are used.

[2] The complete tree would represent a standard Markov Model of order L.

time-steps. After the tree is built, the empirical conditional probability distributions are smoothed by adding a constant probability γ_{min} to all symbols in the alphabet and renormalizing.

3.3 Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) is a two-layer undirected graphical model with a set of visible and a set of hidden units. It is a special, bipartite form of the Boltzmann Machine (Ackley et al., 1985), in which the interaction terms are restricted to units from different layers. The joint distribution over observed and latent variables is defined through an energy function, which assigns a scalar energy to every possible configuration of the variables:

$$P\left(\mathbf{v},\mathbf{h}|\boldsymbol{\theta}\right) = \frac{1}{Z\left(\boldsymbol{\theta}\right)} \exp\left(-E\left(\mathbf{v},\mathbf{h}|\boldsymbol{\theta}\right)\right), \tag{1}$$

where $Z(\boldsymbol{\theta})$ is a normalizing constant called the partition function and $\boldsymbol{\theta}$ is used to denote the set of model parameters.

In its original form, an RBM has binary, logistic units in both layers[3] and its energy function is defined as:

$$E\left(\mathbf{v},\mathbf{h}|\boldsymbol{\theta}\right) = -\mathbf{c}^{\mathsf{T}}\mathbf{v} - \mathbf{b}^{\mathsf{T}}\mathbf{h} - \mathbf{v}^{\mathsf{T}}\mathbf{W}\mathbf{h}, \tag{2}$$

where $\mathbf{c}$ and $\mathbf{b}$ are the biases for the visible and hidden units, respectively, and $\mathbf{W}$ is the weight matrix for the interaction terms.

Inference in this model can be performed efficiently using block Gibbs sampling, as due to the bipartite structure of the model, the conditional distributions of the hidden units given the visibles and of the visible units given the hiddens factorize.

Maximum Likelihood learning in the RBM is difficult due to the partition function $Z(\boldsymbol{\theta})$ which is typically intractable[4]. However, parameter estimation can be performed using Contrastive Divergence (Hinton, 2002), an objective that approximates the likelihood and has been shown to work well in practice.

4 Models

4.1 Dirichlet-VMM

The VMM is similar to an n-gram model in that its performance is significantly influenced by the smoothing technique used. An alternative to a standard form of variable length Markov model is a hierarchical model, where each conditional multinomial distribution in the tree is sampled from a dirichlet Distribution, centered at the sample multinomial for the parent node. In this model smoothing is performed implicitly by taking a Bayesian approach and introducing an appropriate prior distribution at each node while building the tree.

[3] However, see for example Welling et al. (2004) on how to define RBMs with real-valued units.

[4] Computing the partition function involves a sum over all possible configurations of visible and hidden units.

More formally, let $\mathbf{m}\left(\mathbf{x}_t|\mathbf{x}_{t-1},\ldots,\mathbf{x}_{t-\tau-1}\right)$ be defined by

$$m_k = P\left(\mathbf{x}_t = k|\mathbf{x}_{t-1},\ldots,\mathbf{x}_{t-\tau-1}\right) \quad .$$

Then we model each conditional distribution as:

$$P\left(\mathbf{x}_t|\mathbf{x}_{t-1},\ldots,\mathbf{x}_{t-\tau}\right) \sim \text{Dirichlet}\left(\alpha\mathbf{m}\left(\mathbf{x}_t|\mathbf{x}_{t-1},\ldots,\mathbf{x}_{t-\tau-1}\right)\right) \quad . \tag{3}$$

This forms a hierarchical tree with the marginal distribution $P(\mathbf{x}_t)$ as the root node, and successively more specific conditional distributions as we traverse down the tree. The intermediate nodes, though identified with particular distribution, are not used directly to model the data; that is done by the leaf nodes.

Learning this hierarchical distribution involves learning the posterior distributions at each level of the hierarchy from the data associated with the given node (i.e. the data that satisfies the conditional distribution).

$$\begin{aligned} &P(\mathbf{x}_t|\mathbf{x}_{t-1},\ldots,\mathbf{x}_{t-\tau},D) \sim \\ &\quad \text{Dirichlet}(\alpha E\left[\mathbf{m}(\mathbf{x}_t|\mathbf{x}_{t-1},\ldots,\mathbf{x}_{t-\tau-1},D)\right] + \mathbf{c}(\mathbf{x}_t,\mathbf{x}_{t-1},\ldots,\mathbf{x}_{t-\tau})), \end{aligned} \tag{4}$$

where the $c_k(\mathbf{x})$ function counts the number of occurrances of sequence $\mathbf{x}$ in the dataset where the last element is in state k, and E denotes expectation.

The mean of the posterior Dirichlet at each node is the prior Dirichlet for the data at the child nodes. Note the top levels of the hierarchy have a large amount of associated data, but as we progress down the tree the amount of data reduces. In the limit where there is no data the posterior distribution for that node is just given by the posterior for the parent node.

This model is directly related to the sequence memoizer (Wood et al., 2009), but is a finite model using Dirichlet distributions, instead of a Pitman Yor model. Using Dirichlet distributions makes the inference procedure entirely conjugate and thus no sampling is required. We call this model a Dirichlet-VMM in this paper.

4.2 Time Convolutional RBM

We propose a Time Convolutional RBM (TC-RBM) as a new way of modeling sequential data with an RBM type network. We believe that models based on the RBM are particularly suitable for capturing the componential structure of music, as they can learn distributed representations of the input space, decoupling the different factors of variation into features being "on" or "off". The TC-RBM is an adaptation of the Convolutional RBM for sequences and it is motivated by the successful application of such models in static image data (Lee et al., 2009; Norouzi et al., 2009).

Previous RBM approaches to sequence modeling use the RBM to model a single time-step and attempt to capture the temporal relations in the data by introducing different types of directed connections from units in previous time-steps (Taylor et al., 2007; Sutskever and Hinton, 2007; Taylor and Hinton, 2009). On the contrary, the TC-RBM is a fully undirected network and attempts to capture the structure of music at a motif level rather than a single time-step.

The TC-RBM is depicted in Fig. 1. Local temporal dependencies are captured by learning an RBM on visible subsequences of fixed length - instead of single data points.

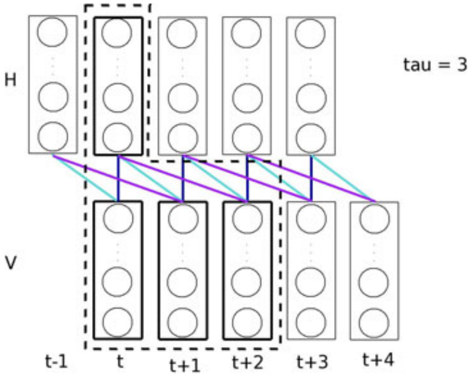


Fig. 1. A Time Convolutional RBM with filter size $\tau = 3$. The dashed frame shows the connections to the hidden units in a single time-step. Each unit receives input from all visible units in a subsequence of length τ. The model is 'unrolled' in time through a weight sharing mechanism.

This allows the hidden units to learn valid configurations for a whole subsequence and thus capture frequent motifs and their transformations. Longer sequences are modelled by applying convolution along time. This weight sharing mechanism allows us to better model boundary effects and provides the model with translation invariance along time, which is desirable as motifs can appear anywhere in a musical piece.

The energy function of the TC-RBM is defined as:

$$E\left(\mathbf{V}, \mathbf{H} | \boldsymbol{\theta}\right) = -\sum_{t} \left(\mathbf{c}^{\mathsf{T}} \mathbf{v}_t + \mathbf{b}^{\mathsf{T}} \mathbf{h}_t + \sum_{k=0}^{\tau-1} \mathbf{v}_{t+k}^{\mathsf{T}} \mathbf{W}_k \mathbf{h}_t \right), \tag{5}$$

where $\mathbf{V}$ is a visible sequence, $\mathbf{H}$ is the hidden configuration for that sequence and τ is the size of the filter we apply[5]. The interaction terms are parameterized by the weight tensor[6] $\mathbf{W}$ and the unit biases, $\mathbf{c}$ and $\mathbf{b}$ for visible and hidden units respectively, are the same for all time-steps.

Similarly to an RBM, the joint probability distribution of the observed and hidden sequence under the TC-RBM is defined as $P(\mathbf{V}, \mathbf{H}|\boldsymbol{\theta}) = \exp\left(-E(\mathbf{V}, \mathbf{H}|\boldsymbol{\theta})\right) / Z(\boldsymbol{\theta})$.

The conditional probability distributions of this model factorize over time and units and are given by softmax and logistic functions:

$$P(v_{i,t} = 1|\mathbf{H}) = \frac{\exp\left(c_i + \sum_{k=0}^{\tau-1} \mathbf{W}_{i,\cdot,k} \mathbf{h}_{t-k}\right)}{\sum_{q} \exp\left(c_q + \sum_{k=0}^{\tau-1} \mathbf{W}_{q,\cdot,k} \mathbf{h}_{t-k}\right)}, \tag{6}$$

[5] The filter size is the number of visible time-steps that a hidden unit receives input from.

[6] Each slice k of the $\mathbf{W}$ tensor is the weight matrix for the connections of hidden units at time t with the visible units at time $t + k$.

$$P(h_{j,t}=1|\mathbf{V}) = \left[1+\exp\left(-b_j - \sum_{k=0}^{\tau-1}\mathbf{v}_{t+k}^{\mathsf{T}}\mathbf{W}_{\cdot,j,k}\right)\right]^{-1} . \tag{7}$$

Inference can be performed using block Gibbs sampling. The computation of (6) and (7) can be performed efficiently by convolving along the time dimension the appropriate slice of the weight tensor with the hidden and visible sequence respectively. As in the RBM, learning can be performed using the Contrastive Divergence rule.

5 Experiments

In the following section we want to assess the ability of the models to learn the inherent structure of melodic sequences belonging to the same genre. An appropriate measure for this evaluation is the marginal likelihood of the data under each model M, $P(\mathbf{D}|M) = \int P(\mathbf{D}|\boldsymbol{\theta}, M)P(\boldsymbol{\theta}|M)d\boldsymbol{\theta}$. However, computing the marginal likelihood under the TC-RBM is intractable[7] and thus we need to make use of other quantitative measures.

In the music modeling literature, evaluation is primarily based on qualitative analysis, like listening to model generations. To our knowledge, the only quantitative measures used so far are next-step prediction accuracy (Paiement, 2008; Lavrenko and Pickens, 2003) and perplexity (Lavrenko and Pickens, 2003). In this work, we broaden this evaluation framework to consider longer future prediction, instead of only next-step, as this provides an insight regarding model performance through time.

To make our comparative analysis more rigorous, we also examine the short order statistics of the models and compare them with the data statistics. To perform this analysis we compute the Kullback-Leibler divergence between the frequency distribution of events in test sequences and in model samples, which measures how well the model statistics match the data, or put differently, how much a model has yet to learn.

Besides the quantitative evaluation, we are also interested in assessing the capabilities of the models to identify and represent the statistical regularities of the data. In the VMM models, the learned lexicon of phrases determines the frequent musical motifs, but does not provide any information regarding the underlying structure, as the encoded patterns are fixed. On the other hand, the TC-RBM learns a distributed representation of the input space; a set of latent features that are ‘on’ or ‘off’ depending on the input signal. We demonstrate that these features are music descriptors extracted from the data and convey information regarding music components such as scale, octaves and chords.

5.1 Data Processing and Representation

In the following experiments we use a dataset comprising 117 traditional reels collected from the Nottingham Folk Music Database[8]. Reels are traditional Scottish and Irish tunes used to accompany dances. All tunes are in the G major scale and have 4/4 meter.

[7] Computing the data likelihood $P(\mathbf{D}|\boldsymbol{\theta}, M)$ in the RHS involves a sum over all possible configurations of visible and hidden units.

[8] We use the MIDI toolbox (Eerola and Toiviainen, 2004) to read and write MIDI files.

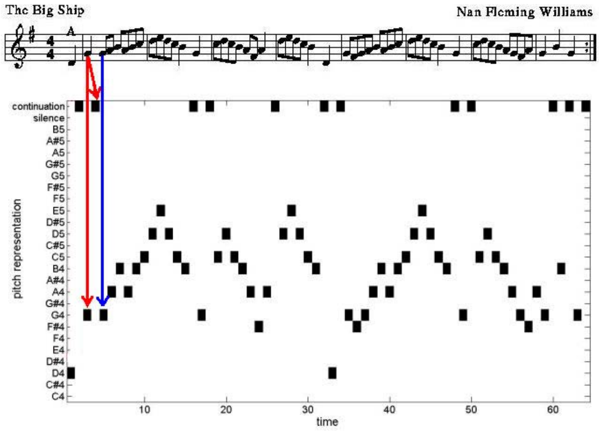


Fig. 2. Data Representation: Time is discretized in eighth note intervals. At each time-step, pitch is represented by a 1-of-26 dimensional vector. Red (left) arrows: a $G4$ quarter note lasts for two time-steps and is represented by $G4$ followed by 'continuation'. Blue (right) arrow: a $G4$ eighth note lasts for one time-step and is represented by a single $G4$.

Our representation is depicted in Fig. 2. The components we wish to model are pitch and duration of the notes in the melody. Duration is modelled implicitly by discretizing time in eighth-note intervals. At each time-step, pitch is encoded using a 1-of-m vector. We use only two octaves, $C4$-$B5$, giving rise to a 24-dimensional vector. Values outside this octave range are trunctated to the nearest octave.

Finally, we augment the 1-of-m vector with two more values. The first one is used to represent music silence. The second one is used to represent 'continuation' of an event and allows us to keep more accurate information concerning the duration of notes.

5.2 Implementation Details

We trained a VMM, a Dirichlet-VMM and a TC-RBM. To set the parameters c_{min}, ϵ_{min} and γ_{min} of the VMM we applied grid search over the product space of the parameters and chose the values that maximize the data log-likelihood using leave-one-out cross validation on the training data. We used the same grid search procedure to set the parameter α of the Dirichlet prior in the Dirichlet-VMM[9].

For the TC-RBM, we used 50 hidden units. We chose the size of the filters to be 8 time-steps, which corresponds to the length of a music bar. For learning the model we used the following settings: CD-5, 500 epochs, 0.5 learning rate decreasing on a

[9] In the VMM, the maximum length L was set to a very large value (100), which resulted in the depth of the tree being controlled by the parameter c_{min} for the frequency counts. The resulting depth for the optimal tree is 13. In the Dirichlet-VMM, we used a global α parameter and applied grid search over the product space of c_{min}, ϵ_{min} and α.

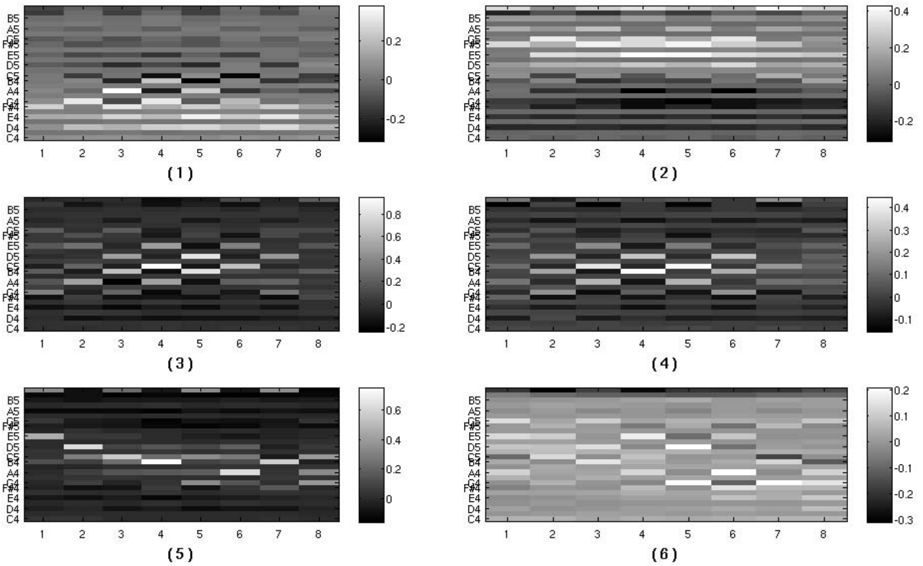


Fig. 3. Weight filters for 6 different hidden units of the learned TC-RBM. All units prefer notes from the G major scale to be ‘on’; these notes are explicitly ticked in the y-axis. Filters 1 and 2 respond to similar patterns, but operate in the lower and higher octave respectively. Filters 3 and 4 respond to notes from either the $Gmaj$ or the Am chord in alternate time-steps. Filter 5 is highly selective to a specific motif, whereas filter 6 responds to several configurations of the scale notes.

fixed schedule, 0.0002 weight decay. We additionally used a sparsity term[10] in the objective function, which encourages hidden units to be ‘off’. We implemented sparsity as described in Lee et al. (2008), and set the desired activity level to 0.1.

5.3 Learning Musical Features

In the TC-RBM each hidden unit is connected with all the visible units from eight subsequent time-steps. This gives rise to a 26×8-dimensional filter for each hidden unit[11].

The filters corresponding to 6 different hidden units from the learned TC-RBM are depicted in Fig. 3. We can notice that all units prefer visible configurations with notes from the G major scale[12] to be ‘on’, but have various degrees of selectivity and respond to different subsets of these notes in different positions.

For instance, filter (6) is fairly broad and may respond to several different configurations of notes from the G major scale, whereas filter (5) is highly selective, responding

[10] It has been suggested (Lee et al., 2009; Norouzi et al., 2009) that due to the over-complete hidden representation of convolutional RBMs, encouraging sparsity is important and can facilitate learning.

[11] The filter for hidden unit j is the slice $\mathbf{W}_{\cdot,j,\cdot}$ of the weight tensor.

[12] Notes from the G major scale: $GABCDEF^{\#}G$.

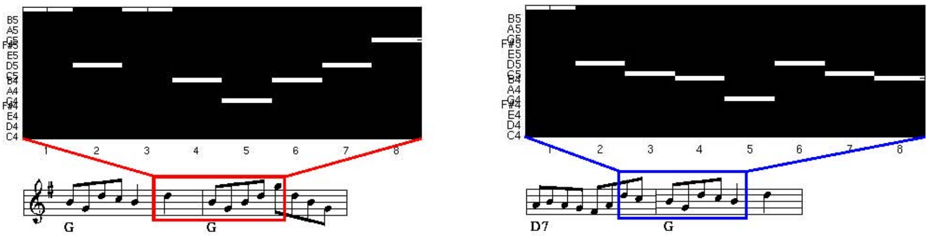


Fig. 4. Two different visible configurations that frequently turn 'on' the hidden unit corresponding to filter (5) from Figure 3 during sampling. The visible configurations are also depicted in the musical score. Each configuration contains a different variation of the motif $D*BG$, which is prominent in filter (5).

primarily to the downwards-upwards movement $EDCBGAB$ through the scale and certain variations of it.

An interesting property of the top two filters is their relation with respect to the octave. Both units respond to similar music phrases. For instance, both units respond to the motif $F^{\#}GAB$ starting at either position 1 or 3. However, the left unit operates in the lower octave ($C4$-$B4$), whereas the right one operates in the higher octave ($C5$-$B5$).

Another interesting property is the relation of the filters to the tone chords of the scale[13]. In several filters, the prefered subset of notes at each time-step corresponds to the notes of a tone chord. This property is particularly prominent in filters (3) and (4). For instance in filter (3), the prefered subset of notes at odd time-steps corresponds to the notes of the $Gmaj$ chord (GBD), whereas at even time-steps to the notes of the Am chord (ACE).

In order to better understand how the filters behave, we looked at random visible configurations that tend to activate a hidden unit during sampling. Figure 4 shows two such visible configurations for the hidden unit corresponding to filter (5). Although the two configurations seem fairly different, they both contain the motif $D*BG$ in positions 2 to 5 with either a pass through A or 'continuation' of D in position 3. Filter (5) is highly responsive to this motif, and although time-steps 6 to 8 in the visible configurations are not highly preferable, the unit is still very likely to turn 'on'.

Overall, we can see that the learned filters encode familiar musical movements, such as arpeggios and scales[14]. However, the interesting and potentially powerful characteristic of the TC-RBM representation is that it also encodes and groups together many possible transformations of these movements. Therefore, in contrast to the VMM representation, the motifs learned by the TC-RBM are not fixed; the TC-RBM filters group together musically sound variations of motifs, thus encoding possible note substitutions, merging and splitting. This is very advantageous when modeling music, given its highly complex and ingenious nature, and also allows for more genuine music generations.

[13] These are chords of three, four or five notes built from alternate scale notes of G Major.

[14] These can be loosely defined as groups of subsequent scale notes, either going up or going down.

5.4 Prediction Task

Given an observed test subsequence we want to evaluate how well a model can predict the following k time-steps. We define the prediction log-likelihood of a test sequence D under a model M with parameters $\boldsymbol{\theta}$, as the log probability of the actual future time-step $d_{t+\tau}$ given time-steps d_1 up to d_t, averaged over all time-steps T_n of the test sequence. More specifically:

$$\log L_\tau(\boldsymbol{\theta}, M; \mathbf{D}) = \frac{1}{T_n} \sum_{t}^{T} \log P(d_{t+\tau} | d_1, \ldots, d_t, \boldsymbol{\theta}, M) \ . \tag{8}$$

We use the empirical marginal distribution[15] as a baseline for evaluating model performance.

Computing Prediction $\log L$ under the VMM and the Dirichlet-VMM. For $\tau = 1$ we can compute (8) exactly under the VMM models. For $\tau > 1$ we need to marginalize over the future time-steps that are between d_t and $d_{t+\tau}$, ie:

$$P\left(d_{t+\tau} | d_1, \ldots, d_t, \boldsymbol{\theta}, M\right) = \sum_{d_{t+1}, \ldots, d_{t+\tau-1}} P(d_{t+1}, \ldots, d_{t+\tau} | d_1, \ldots, d_t, \boldsymbol{\theta}, M) \ . \tag{9}$$

We approximate this distribution by drawing a number of sampled paths from the VMM and averaging over the conditional probability distributions defined by these paths which are given exactly under the VMM:

$$P(d_{t+\tau} | d_1, \ldots, d_t, \boldsymbol{\theta}, M) \approx \frac{1}{S} \sum_{s=1}^{S} P(d_{t+\tau} | d_1, \ldots, d_t, d^s_{t+1}, \ldots, d^s_{t+\tau-1}, \boldsymbol{\theta}, M). \tag{10}$$

We use 100 sampled paths in the experiments reported here.

Computing Prediction $\log L$ under the TC-RBM. In order to evaluate (8) under the TC-RBM, we need to marginalize over future visible time-steps that are between d_t and $d_{t+\tau}$ for $\tau > 1$ and over the possible configurations of hidden units for time-steps t to $t+\tau$. To avoid this computation we approximate the predictive distribution using samples from the model. The sampling procedure is given in Algorithm 1.

In our experiments, we use 100 chains and run 15 Gibbs iterations within each chain. Overall, we use 500 samples to approximate the predictive distribution, discarding the first 10 samples from each chain.

Results. Figure 5 shows the log-likelihood of predicting the true succession given an observed sub-sequence from a test tune under different models. As already mentioned, our baseline for assessing model performance is the empirical marginal distribution. The log-likelihood of the test data under the empirical marginal corresponds to the black curve.

[15] The empirical distribution of the training data under the assumption that all time-steps are iid (independently and identically distributed). This distribution is the best predictor in the absence of temporal dependencies.

Algorithm 1. Sampling Procedure for the TC-RBM

```
Let $\mathbf{V}$ be a visible sequence and $\mathbf{H}$ a hidden sequence
Initialize $\mathbf{V}$ randomly
Set $\mathbf{v}_{1...t} = d^n_{1...t}$
While $s <$ numberOfSamples
    $\mathbf{H} \sim P_{TC-RBM}(\mathbf{H}|\mathbf{V}, \boldsymbol{\theta})$    (Equation 7)
    $\mathbf{V} \sim P_{TC-RBM}(\mathbf{V}|\mathbf{H}, \boldsymbol{\theta})$    (Equation 6)
    Clamp to context: $\mathbf{v}_{1...t} = d^n_{1...t}$
    If equilibrium reached
        $\mathbf{H}^s = \mathbf{H}$
        $s = s + 1$
    end
end
$P(d^n_{t+\tau}|d^n_{1...t}, \boldsymbol{\theta}, M) = \frac{1}{S}\sum P_{TC-RBM}(\mathbf{v}_{t+\tau}|\mathbf{H}^s, \boldsymbol{\theta})$
```

Compared to the empirical marginal distribution, the standard VMM (green x) performs significantly better in predicting the first two future time-steps, only slightly better for time-steps 3 and 4 and significantly worse than the empirical marginal after the 5th time-step.

Both the Dirichlet-VMM (cyan crosses) and the TC-RBM (blue stars) perform significantly better than the standrad VMM in predicting all future time-steps. These two models have similar performance in the prediction task, with the Dirichlet-VMM outperforming the TC-RBM for the first two time-steps and their prediction log-likelihood being almost the same from the 3rd time-step onwards.

We should note that the performance of the TC-RBM in prediction may be compromised by the fact that the block Gibbs procedure samples the future sub-sequence as a whole at each iteration. This means that due to the convolutional structure of the model, the time-step we are trying to predict receives information not only from the past, which is clamped to the observed context, but also from the future which is initialized randomly and can thus drive the samples into different energy basins.

Compared to the empirical marginal distribution, both the Dirichlet-VMM and the TC-RBM perform better for the first 10 time-steps. The prediction log-likelihood under the models is initially much higher than the one under the empirical marginal distribution, but decays as we try to predict further into the future. The models slowly forget the information upon which they have been conditioned and after the 10th time-step converge to a steady-state distribution, which is slightly worse than the empirical marginal distribution for prediction.

While long-term prediction is useful for characterizing model behaviour through time, it is not adequate for evaluating the generative capabilities of the models. For instance, even if a musical phrase is highly predictable given a certain context, the models can get bad predictive performance if they are not able to determine the correct starting time-step for the phrase.

Nevertheless, we can note that in contrast to the standard VMM, our proposed models converge to the empirical marginal distribution over time and thus are better in

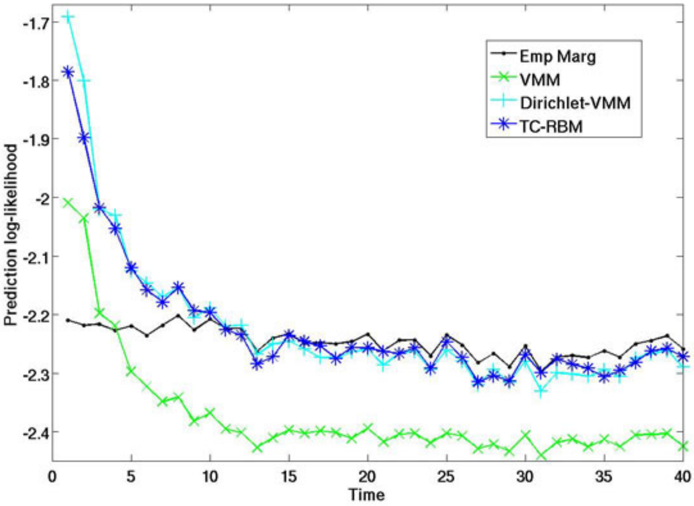


Fig. 5. pREDICTION Log-Likelihood under different models plotted as a function of time. The Log-Likelihood is averaged across 2,000 configurations of context-future observations, randomly selected from the test data.

capturing the statistical regularities in the data, which is the first step towards realistic music generation.

5.5 Using the Kullback-Leibler Divergence to Compare Statistics

The Kullback-Leibler (KL) divergence is a measure of how different two probability distributions, P and Q, are. For discrete random variables, it is defined as $D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$ and shows the average number of extra bits needed to encode events from a distribution P with a code based on an approximating distribution Q. If the true distribution that generated the data is P and the model distribution is Q, then the lower the KL-divergence the better the model matches the data.

To compare model statistics with data statistics, we compute the frequency distribution of events in samples generated by each of the models and in test sequences, and compute the KL-divergence between the normalized data and model frequencies. More specifically, let d_t denote the observation of a single time-step at time t. Then to compare first-order statistics we estimate the KL-divergence between $P(d_t)$ and $Q(d_t)$ by computing:

$$D_{\mathrm{KL}}(P_{\mathrm{data}}(d_t)\|Q_M(d_t)) = \frac{1}{N}\sum_{n=1}^{N} P_{\mathrm{data}}(d_t^n) \log \frac{P_{\mathrm{data}}(d_t^n)}{Q_M(d_t^n)}, \tag{11}$$

where $P_{\mathrm{data}}(d_t)$ is the empirical marginal distribution of data sequences and $Q_M(d_t)$ is the marginal distribution of samples generated by model M. Similarly, for pairwise

Table 1. KL-divergence between data statistics and model statistics. We report the mean and variance (in the parenthesis) of the KL-divergence for each statistic. These are computed using 50 resamples, obtained by random sampling with replacement from the test dataset.

	order 1	order 2	order 3	order 4	order 5	order 6
Trainset	0.032 (1*e*-4)	0.433 (0.012)	1.351 (0.093)	3.150 (0.197)	5.455 (0.330)	7.985 (0.479)
TC-RBM	0.064 (2*e*-4)	0.273 (4*e*-4)	0.872 (0.002)	2.420 (0.047)	5.244 (0.248)	8.584 (0.645)
Dir-VMM	0.045 (3*e*-4)	0.302 (0.005)	1.158 (0.076)	2.594 (0.172)	4.295 (0.357)	6.462 (0.672)
VMM	0.187 (1*e*-4)	0.481 (4*e*-4)	1.331 (0.023)	3.242 (0.114)	5.839 (0.284)	8.772 (0.452)
	lag 1	lag 2	lag 3	lag 4	lag 5	lag 6
Trainset	0.228 (0.002)	0.251 (0.003)	0.188 (8*e*-4)	0.236 (0.003)	0.180 (8*e*-4)	0.254 (0.001)
TC-RBM	0.229 (6*e*-4)	0.203 (5*e*-4)	0.222 (6*e*-4)	0.201 (5*e*-4)	0.204 (9*e*-4)	0.175 (4*e*-4)
Dir-VMM	0.198 (0.001)	0.175 (0.001)	0.224 (0.001)	0.184 (0.002)	0.215 (0.001)	0.202 (0.001)
VMM	0.476 (2*e*-4)	0.474 (4*e*-4)	0.542 (4*e*-4)	0.477 (6*e*-4)	0.533 (5*e*-4)	0.472 (6*e*-4)

statistics we compute $D_{\mathrm{KL}}(P_{\mathrm{data}}(d_t, d_{t+1}) \| Q_M(d_t, d_{t+1}))$, for third order statistics $D_{\mathrm{KL}}(P_{\mathrm{data}}(d_t, d_{t+1}, d_{t+2}) \| Q_M(d_t, d_{t+1}, d_{t+2}))$, and so on.

Since the true distribution that generated the data is unknown, we perform a bootstrapping procedure for the estimation of the KL-divergence. More specifically, we compute the KL-divergence for each statistic 50 times, each time using a different data resample, obtained by random sampling with replacement from the original test dataset. In our results, we report the mean and variance of the KL-divergence for each statistic.

The number of possible events grows exponentially with the order we consider, which makes the statistics for higher-orders less reliable, given that we have a finite test set. In order to get a better understanding of how the models perform through time, we additionally consider pairwise statistics with lags, that is statistics of events comprising two time-steps which are not adjacent in time. For instance for lag $l = 1$ we consider the frequencies of events (d_t, d_{t+2}), for lag $l = 2$ we consider (d_t, d_{t+3}) and so on.

Results. Table 1 shows the mean and variance of the KL-divergence between the statistics of test sequences and a priori samples for various models. The first row compares test sequences to train sequences and is used as a reference for interpreting the results. Looking at the first order statistics we can note that the TC-RBM and the Dirichlet-VMM have much lower KL-divergence than the VMM, with the Dirichlet-VMM having the lowest amongst the models. In fact, the KL-divergence for the former two models is very close to the KL-divergence between test sequences and train sequences, which indicates that samples generated from these models match the statistics of the test data well.

For the second, third and fourth order statistics, the TC-RBM has the lowest KL-divergence, with the Dirichlet-VMM following closely and the VMM lagging behind. Interestingly, the KL-divergence of these statistics for the TC-RBM and the Dirichlet-VMM is even lower than the one for the train data. We believe that this stems from the fact that the models are capturing the underlying structure that characterizes the whole musical genre, and to some extent ignore the finer structure that characterizes each individual music piece. This can result in model samples that have higher inter- and lower intra-piece similarity than a set of real music sequences.

For fifth and sixth order statistics, the KL-divergence for the TC-RBM and the VMM is close to the KL-divergence for the train data, whereas for the Dirichlet-VMM is lower. As mentioned earlier, the estimates for higher order statistics are less reliable, since the number of possible configurations is exponentially large and thus very difficult to characterize from a finite set of samples.

Finally, for the pairwise statistics with lags, the KL-divergence for both the TC-RBM and the Dirichlet-VMM is low, very close to the one for the train data, whereas for the VMM it is considerably higher. This suggests that our proposed methods respect the short order statistics of the musical genre and are better than the VMM in capturing the statistical regularities of the data through time.

6 Discussion

We addressed the problem of learning a generative model for music melody by considering two probabilistic models, the Dirichlet-VMM and the Time Convolutional RBM. We showed that the TC-RBM, trained on a dataset of tunes from the same genre, learns descriptive musical features that can be used to decompose the underlying structure of the data into musical components such as scale, octave and chord.

We performed a comparative analysis of the two models with the standard VMM, which, to our knowledge is state of the art in melody generation. We showed that in a long-term prediction task both models perform significantly better than the VMM and comparably with each other. The Dirichlet-VMM is a better next-step predictor, which can be partially accredited to its main strength, that is its ability to use shorter or longer contexts depending on whether they provide useful information or not.

We evaluated the short order statistics of the models by comparing the Kullback-Leibler divergence between test sequences and model samples. We demonstrated that sampled generations from our proposed methods match the statistics of the test sequences considerably better than samples from the VMM and respect the genre statistics, as the KL-divergence for the TC-RBM and the Dirichlet-VMM is very close to the KL-divergence between test and train sequences.

The ability of the TC-RBM to extract descriptive music features allows us to consider hierarchical approaches for melody generation, which can help modulate the appearance of features through time. We are currently experimenting with deeper TC-RBM architectures, where TC-RBMs are stacked on top of one another in a greedy manner (see Hinton et al. (2006) for the RBM case). Deep models have been shown to learn hierarchical representations of the input space, where more abstract features are captured in higher layers, which according to the tonal music theory (Lerdahl and Jackendoff, 1983) is how music composition should be understood.

Finally, an interesting direction for future research in music modeling involves exploration of methods that can distinguish between inter- and intra-piece similarity. The methods examinded in this work can learn the statistical relations within a musical genre, but are not able to effectively model piece-wise variation. Considering methods that enable us to sample a prior distribution for each piece, such as topic models, would be a first step towards this direction.

Acknowledgements. Athina Spiliopoulou is partly funded by an EPSRC scholarship.

References

Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cognitive Science 9(1), 147–169 (1985)

Dubnov, S., Assayag, G., Lartillot, O., Bejerano, G.: Using machine-learning methods for musical style modeling. Computer 36(10), 73–80 (2003)

Eck, D., Lapalme, J.: Learning musical structure directly from sequences of music. Technical report, Université de Montreal (2008)

Eck, D., Schmidhuber, J.: Learning the long-term structure of the blues. In: Dorronsoro, J.R. (ed.) ICANN 2002. LNCS, vol. 2415, pp. 284–289. Springer, Heidelberg (2002)

Eerola, T., Toiviainen, P.: MIDI Toolbox: MATLAB Tools for Music Research. University of Jyväskylä, Jyväskylä, Finland (2004), www.jyu.fi/musica/miditoolbox/

Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Computation 14(8), 1771–1800 (2002)

Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation 18(7), 1527–1554 (2006)

Lavrenko, V., Pickens, J.: Polyphonic music modeling with random fields. In: Rowe, L.A., Vin, H.M., Plagemann, T., Shenoy, P.J., Smith, J.R. (eds.) Proceedings of the Eleventh ACM International Conference on Multimedia, ACM Multimedia, pp. 120–129. ACM, New York (2003)

Lee, H., Ekanadham, C., Ng, A.Y.: Sparse deep belief net model for visual area V2. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (eds.) NIPS. Advances in NIPS, vol. 20. MIT Press, Cambridge (2008)

Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) ICML. ACM ICPS, vol. 382, p. 77. ACM, New York (2009)

Lerdahl, F., Jackendoff, R.: A Generative Theory of Tonal Music. The MIT Press, Cambridge (1983)

Norouzi, M., Ranjbar, M., Mori, G.: Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning. In: IEEE Computer Society Conference on CVPR, CVRP 2009, pp. 2735–2742. IEEE, Los Alamitos (2009)

Paiement, J.-F.: Probabilistic Models for Music. PhD thesis, Ecole Polytechnique Fédérale de Lausanne (EPFL) (2008)

Ron, D., Singer, Y., Tishby, N.: The power of amnesia. In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) NIPS. Advances in NIPS, vol. 6, pp. 176–183. Morgan Kaufmann, San Francisco (1994)

Sutskever, I., Hinton, G.E.: Learning multilevel distributed representations for high-dimensional sequences. Journal of ML Research - Proceedings Track 2, 548–555 (2007)

Taylor, G.W., Hinton, G.E.: Factored conditional restricted Boltzmann machines for modeling motion style. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) ICML. ACM ICPS, vol. 382, p. 129. ACM, New York (2009)

Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) NIPS. Advances in NIPS, vol. 19, pp. 1345–1352. MIT Press, Cambridge (2007)

Weiland, M., Smaill, A., Nelson, P.: Learning musical pitch structures with hierarchical hidden Markov models. Technical report, University of Edinburgh (2005)

Welling, M., Rosen-Zvi, M., Hinton, G.E.: Exponential family harmoniums with an application to information retrieval. In: NIPS. Advances in NIPS, vol. 17 (2004)

Wood, F., Archambeau, C., Gasthaus, J., James, L., Teh, Y.W.: A stochastic memoizer for sequence data. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) ICML. ACM ICPS, vol. 382, p. 142. ACM, New York (2009)

Fast Projections onto $\ell_{1,q}$-Norm Balls for Grouped Feature Selection

Suvrit Sra

MPI for Intellingent Systems
72076 Tübingen, Germany

Abstract. Joint sparsity is widely acknowledged as a powerful structural cue for performing feature selection in setups where variables are expected to demonstrate "grouped" behavior. Such grouped behavior is commonly modeled by Group-Lasso or Multitask Lasso-type problems, where feature selection is effected via $\ell_{1,q}$-mixed-norms. Several particular formulations for modeling groupwise sparsity have received substantial attention in the literature; and in some cases, efficient algorithms are also available. Surprisingly, for *constrained* formulations of fundamental importance (e.g., regression with an $\ell_{1,\infty}$-norm constraint), highly scalable methods seem to be missing. We address this deficiency by presenting a method based on spectral projected-gradient (SPG) that can tackle $\ell_{1,q}$-constrained convex regression problems. The most crucial component of our method is an algorithm for projecting onto $\ell_{1,q}$-norm balls. We present several numerical results which show that our methods attain up to 30X speedups on large $\ell_{1,\infty}$-multitask lasso problems. Even more dramatic are the gains for just the $\ell_{1,\infty}$-projection subproblem: we observe almost three orders of magnitude speedups compared against the currently standard method.

1 Introduction

Sparsity offers powerful structural information that enables recovering unknown, high-dimensional vectors robustly. Consequently, sparsity has been intensively studied in signal processing, machine learning, and statistics, where it has played a key role in numerous algorithms and applications. The associated literature has grown too large, and a summary will be futile, so we refer the reader to [2, 24, 30, 31] as starting points.

Typically sparsity constrained problems that arise in machine learning and statistics may be written as instances of the following general problem:

$$\min_{\boldsymbol{w}} \quad \mathcal{L}(\boldsymbol{w}) + \lambda R(\boldsymbol{w}), \tag{1}$$

where $\mathcal{L}$ is differentiable, convex loss-function, $\lambda > 0$ is a scalar, while R is a convex (possibly nonsmooth) regularizer that models sparsity. Alternatively, one can consider the following constrained formulation of (1):

$$\min_{\boldsymbol{w}} \quad \mathcal{L}(\boldsymbol{w}) \quad \text{s.t.} \quad R(\boldsymbol{w}) \le \gamma, \tag{2}$$

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 305–317, 2011.

for an appropriate scalar $\gamma > 0$. We focus on the latter formulation, especially because it is particularly amenable to a simple first-order optimization procedure. We note another benefit that can make formulation (2) attractive: several variants of gradient-projection [6] remain applicable even when $\mathcal{L}$ is not convex; mere differentiability suffices. Moreover, theoretical analysis of (2) is simpler because the constraint ensures that we are minimizing over a compact set.

Amongst the dizzying number of variants of (2) that have been studied the literature, we consider in this paper a particular family: *groupwise sparsity*. Here, the regularizer R selects (or ignores) entire groups of variables simultaneously, e.g. in multitask learning [12, 13, 19, 26], or in Group-Lasso [3, 36, 37]. One practical way to induce groupwise sparsity is to let R be a mixed-norm, defined as follows (for a more general definition, see [39]). Let a vector $\boldsymbol{w} \in \mathbb{R}^d$ be partitioned into the subvectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_G$, where $\boldsymbol{w}_g \in \mathbb{R}^{d_g}$ for $1 \leq g \leq G$, and let $q \geq 1$. Then, the $\ell_{1,q}$ *mixed-norm* for $\boldsymbol{w}$ is given by

$$\|\boldsymbol{w}\|_{1,q} = \sum\nolimits_{g=1}^{G} \|\boldsymbol{w}_g\|_q. \tag{3}$$

The most common variants of (3) are $\|\cdot\|_{1,2}$ and $\|\cdot\|_{1,\infty}$; the former is often used in Group-Lasso [37], while the latter arises in compressed sensing [35] and multitask Lasso [19]. Less common are versions with $1 < q < \infty$, see e.g., [18, 29, 38]—though both work with penalized formulations. Also note that if $0 < q < 1$, then (3) yields a nonconvex, quasinorm, while for $q = 0$ and $q = 1$, $\|\cdot\|_{1,q}$ totally decouples, thus losing the grouping effect of mixed-norms.

Other authors have considered overlapping versions of (3), i.e., where subvectors $\boldsymbol{w}_g$ might not be disjoint. However, unless the subvectors have a special structure [15, 22, 24], the resulting mixed-norm can be computationally very expensive. Because our aim is on developing fast algorithms, we focus on the non-overlapping version (3), especially because this version is widely applicable and enjoys numerous applications [5, 11, 12, 14, 16, 19, 26, 34, 36].

Contributions. Before beginning our theoretical discussion, we enlist the key contributions of this paper here for the reader's convenience:

1. An SPG-based algorithm for convex, sparsity-constrained regression problems (such as lasso, multitask lasso, group lasso, etc.).
2. A root-finding procedure grounded in convex-duality, which is crucial to making the SPG-based algorithm practical. As a byproduct we also obtain an efficient method to tackle operators for ℓ_{∞,q^*}-mixed norms.
3. Experimental validation and illustration of our methods on large-scale multitask lasso, leading to a highly competitive method for it.

2 Algorithm and Theory

The simplest method to solve (2) is perhaps *gradient-projection* [6], where starting with some $\boldsymbol{w}^0$, one iterates

$$\boldsymbol{w}^{t+1} = \operatorname{proj}(\boldsymbol{w}^t - \alpha^t \nabla\mathcal{L}(\boldsymbol{w}^t), \gamma, q), \quad t = 0, 1, \ldots, \tag{4}$$

where proj is the *projection operator*, defined as

$$\operatorname{proj}(\boldsymbol{v}, \gamma, q) := \operatorname{argmin}_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{u}\|_{1,q} \leq \gamma. \tag{5}$$

To implement (4) efficiently, there are three obvious, key components: (i) the gradient $\nabla\mathcal{L}$; (ii) the *step-size* $\alpha^t > 0$; and (iii) the projection operator (5). Almost all methods require computation of the gradient (or an approximation).

Let us begin by considering stepsize computation. There exist several classical strategies for computing the step-size α, for example, by exact minimization, by backtracking, Armijo line-search, and so on [6]. However, in general, these strategies are more expensive than absolutely necessary, and although scalable, may even lead to slowly converging algorithm [6, 16, 33].

A fairly recent, and powerful alternative is offered by the *spectral step-sizes* of Barzilai and Borwein [4] (BB), which avoid line-search by providing closed-form formulae for choosing α^t. These stepsizes are

$$\alpha_{BB1} := \frac{\boldsymbol{y}_t^T \boldsymbol{y}_t}{\boldsymbol{s}_t^T \boldsymbol{y}_t}, \quad \text{or} \quad \alpha_{BB2} := \frac{\boldsymbol{y}_t^T \boldsymbol{s}_t}{\boldsymbol{s}_t^T \boldsymbol{s}_t}, \tag{6}$$

where $\boldsymbol{y}_t = \boldsymbol{w}^t - \boldsymbol{w}^{t-1}$, and $\boldsymbol{s}_t = \nabla\mathcal{L}(\boldsymbol{w}^t) - \nabla\mathcal{L}(\boldsymbol{w}^{t-1})$. Directly setting α^t to one of the choices in (6) leads to a gradient-projection method whose convergence is not guaranteed. However, often this substitution displays strong empirical performance. Thus, there have been numerous attempts at using (6) in conjunction with the iteration (4), and one of the most well-known amongst them is the spectral-projected gradient (SPG) method [7].

SPG essentially substitutes (6) in (4) (using a safeguard to ensure bounded and nonzero stepsizes), and thus leverages the strong empirical performance offered by BB stepsizes [4, 7, 9, 33]. SPG ensures global convergence by invoking a nonmontone line search strategy that allows the objective value to occasionally increase, while maintaining some bookkeeping information that allows construction of a descending subsequence.

2.1 Efficiently Computing Projections

The second component of (4) is a core element of our algorithm: projection onto the $\ell_{1,q}$-norm ball. Since every iteration of (4) calls for a projection, it is critical to implement it efficiently. But before we describe the details of our projection algorithm, we first prove a useful duality result.

Lemma 1 (Dual-Norm). *Let $q \geq 1$, and let q^* be its* conjugate exponent*, i.e., it satisfies $1/q + 1/q^* = 1$. Then, the norm $\|\cdot\|_{\infty,q^*}$ is dual to $\|\cdot\|_{1,q}$.*

Proof. By definition, the norm dual to (3) is given by [32]:

$$\|\boldsymbol{u}\|_* := \sup_{\boldsymbol{w}} \; \{\langle \boldsymbol{u}, \, \boldsymbol{w}\rangle \mid \|\boldsymbol{w}\|_{1,q} \leq 1\}. \tag{7}$$

Consider the inequality

$$\langle \boldsymbol{u}, \, \boldsymbol{w}\rangle = \sum\nolimits_{g=1}^{G} \langle \boldsymbol{u}_g, \, \boldsymbol{w}_g\rangle \leq \sum\nolimits_{g=1}^{G} \|\boldsymbol{u}_g\|_{q^*}\|\boldsymbol{w}_g\|_q, \tag{8}$$

which follows directly from Hölder's inequality. Now introduce vectors, $\boldsymbol{\xi} = [\|\boldsymbol{u}_g\|_{q^*}]$, and $\boldsymbol{\psi} = [\|\boldsymbol{w}_g\|_q]$, and apply Hölder's inequality again to obtain

$$\langle \boldsymbol{\xi},\, \boldsymbol{\psi} \rangle \leq \|\boldsymbol{\xi}\|_\infty \|\boldsymbol{\psi}\|_1 = \|\boldsymbol{u}\|_{\infty,q^*} \|\boldsymbol{w}\|_{1,q}, \tag{9}$$

so that from Definition (7) we conclude that $\|\boldsymbol{u}\|_* \leq \|\boldsymbol{u}\|_{\infty,q^*}$. To prove that $\|\boldsymbol{u}\|_* = \|\boldsymbol{u}\|_{\infty,q^*}$, we now show that for each $\boldsymbol{u}$, we can find a $\boldsymbol{w}$ satisfying $\|\boldsymbol{w}\|_{1,q} = 1$, and for which $\langle \boldsymbol{u},\, \boldsymbol{w} \rangle = \|\boldsymbol{u}\|_{\infty,q^*}$. To that end, let g^* be any index in the set $\{\operatorname{argmax}_{1 \leq g \leq G} \|\boldsymbol{u}_g\|_{q^*}\}$. To simplify notation let $\boldsymbol{z} = \boldsymbol{u}_{g^*}$ and $\boldsymbol{y} = \boldsymbol{w}_{g^*}$, and then set $\boldsymbol{w}_g = 0$ for all g except for g^* for which

$$(\boldsymbol{w}_{g^*})_i = \boldsymbol{y}_i = \frac{\operatorname{sgn}(z_i)|z_i|^{q^*-1}}{\|\boldsymbol{z}\|_{q^*}^{q^*-1}}. \tag{10}$$

Now observe that the inner-product $\langle \boldsymbol{u},\, \boldsymbol{w} \rangle$ satisfies

$$\begin{aligned} \langle \boldsymbol{u},\, \boldsymbol{w} \rangle &= \sum\nolimits_g \langle \boldsymbol{u}_g,\, \boldsymbol{w}_g \rangle = \langle \boldsymbol{z},\, \boldsymbol{y} \rangle = \frac{1}{\|\boldsymbol{z}\|_{q^*}^{q^*-1}} \sum\nolimits_i z_i y_i \\ &= \frac{1}{\|\boldsymbol{z}\|_{q^*}^{q^*-1}} \sum\nolimits_i |z_i|^{q^*} = \|\boldsymbol{z}\|_{q^*} = \|\boldsymbol{u}\|_{\infty,q^*}, \end{aligned}$$

and that the norm $\|\boldsymbol{w}\|_{1,q} = \|\boldsymbol{w}_{g^*}\|_q = \|\boldsymbol{y}\|_q$ satisfies

$$\begin{aligned} \|\boldsymbol{y}\|_q &= \big(\sum\nolimits_i |y_i|^q\big)^{1/q} = \Big(\sum\nolimits_i |z_i|^{q(q^*-1)} / \|\boldsymbol{z}\|_{q^*}^{q(q^*-1)}\Big)^{1/q} \\ &= \big(\sum\nolimits_i |z_i|^{q^*} / \|\boldsymbol{z}\|_{q^*}^{q^*}\big)^{1/q} = 1, \end{aligned}$$

which complete the proof of the theorem. □

Note: As the reader may already guess, the proof above also extends to showing the ℓ_{p^*,q^*}-norm is dual to the $\ell_{p,q}$-norm; but we omit details for brevity.

2.2 Projections via Proximity

Often, for computing $\operatorname{proj}(\boldsymbol{v}, \gamma, q)$ it proves beneficial to consider the closely related *proximity operator*:

$$\operatorname{prox}(\boldsymbol{v}, \theta, q) := \operatorname{argmin}_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 + \theta \|\boldsymbol{u}\|_{1,q}. \tag{11}$$

In general, computing (11) can be simpler—for instance, with $q = 1$, (11) is merely the soft-thresholding operator [10], and with $q = \infty$ efficient methods already exist [11, 21]. Interestingly, for $q = 2$ both (11) and (5) are equally easy [5] to compute. The key reason why one may expect (11) to be simpler is because with non-overlapping variable groups, it separates into G independent ℓ_q-norm proximity subtasks. We now show how to leverage this separability.

The idea is simple and could be regarded as well-known [27]. But exploiting it effectively requires some care, as we show below. Let $L(\boldsymbol{u}, \theta)$ be the Lagrangian to (5), and let the dual optimal be denoted by θ^*. Then, assuming strong-duality, the primal optimal $\boldsymbol{u}^*$ that solves (5) is obtained by computing

$$\boldsymbol{u}^*(\theta^*) = \text{argmin}_{\boldsymbol{u}}\ L(\boldsymbol{u}, \theta^*) := \text{argmin}_{\boldsymbol{u}}\ \tfrac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 + \theta^*(\|\boldsymbol{u}\|_{1,q} - \gamma). \tag{12}$$

But computing (12) requires the knowledge of the optimal θ^*, which in turn depends on u^*. How do we break this circularity? The almost obvious, but really crucial observation here is that the optimal θ^* can be computed by solving a single nonlinear equation. Let us see how.

Note that if $\|\boldsymbol{v}\|_{1,q} \leq \gamma$, then $\boldsymbol{u}^* = \boldsymbol{v}$ is the optimal solution. So we assume that $\|\boldsymbol{v}\|_{1,q} > \gamma$; in this case, the optimal θ^* satisfies $\|\boldsymbol{u}(\theta^*)\|_{1,q} = \gamma$. Define now

$$\boldsymbol{u}(\theta) := \text{prox}(\boldsymbol{v}, \theta, q), \tag{13}$$

and consider the nonlinear function

$$g(\theta) = -\gamma + \|\boldsymbol{u}(\theta)\|_{1,q}, \tag{14}$$

from which the optimal θ^* can be obtained by solving $g(\theta) = 0$. Toward solving this equation, Lemma 2 proves useful.

Lemma 2. *Let $g(\theta)$ be defined by* (14)*. There exists an interval $[0, \theta_{\max}]$ on which $g(\theta)$ is monotonically decreasing, and differs in sign at the endpoints.*

Proof. First, observe that by our assumption on $\|\boldsymbol{v}\|_{1,q}$, the inequality $g(0) = -\gamma + \|\boldsymbol{v}\|_{1,q} > 0$ holds.

Next, notice that for $\theta' \geq \|\boldsymbol{v}\|_{\infty,q^*}$, the solution $\boldsymbol{u}(\theta') = 0$. To see why, suppose $\theta' \geq \|\boldsymbol{v}\|_{\infty,q^*}$ but $\boldsymbol{u}(\theta') \neq 0$. Then, $\frac{1}{2}\|\boldsymbol{u}(\theta) - \boldsymbol{v}\|_2^2 + \theta'\|\boldsymbol{u}(\theta)\|_{1,q} < \frac{1}{2}\|\boldsymbol{v}\|_2^2$. Since $\|\cdot\|_2^2$ is strictly convex, for all $\boldsymbol{u}$ we have the inequality $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 - \|\boldsymbol{v}\|_2^2 > -2\langle \boldsymbol{v}, \boldsymbol{u}\rangle$. Combining the two inequalities we obtain $\theta' < \langle \boldsymbol{v}, \boldsymbol{u}(\theta)\rangle / \|\boldsymbol{u}(\theta)\|_{1,q}$. Hence, if $\theta' \geq \sup_{\boldsymbol{u} \neq 0} \langle \boldsymbol{v}, \boldsymbol{u}\rangle / \|\boldsymbol{u}\|_{1,q} = \|\boldsymbol{v}\|_{\infty,q^*}$ (see (7)), then $\boldsymbol{u}(\theta)^*$ must equal 0. This argument implies that $g(\theta') = -\gamma < 0$, allowing us to select $\theta_{\max} = \theta'$.

Finally, to establish monotonicity of $g(\theta)$, merely recognize that $g(\theta)$ is the derivative of the (concave) *dual function* $\inf_{\boldsymbol{u}} L(\boldsymbol{u}, \theta)$. □

Since g changes sign in the interval $[0, \theta_{\max}]$, is continuous, and monotonic, it has a unique root in $[0, \theta_{\max}]$. We can compute this root θ^* to ϵ-accuracy using bisection in $O(\log(\theta_{\max}/\epsilon))$ iterations. In practice, we *do not use* plain bisection, but invoke a more powerful root-finder that combines bisection, inverse quadratic interpolation, and the secant method. Algorithm 1 provides pseudocode for computing the projection (5) based on the above ideas.

2.3 Computing the Proximity Operators

Now that we have reduced $\ell_{1,q}$-projections to a sequence of proximity computations, a few words about the associated proximity operators are in order. Depending on the choice of q, these operators can range from trivial to complicated. However, for all the computations, the key benefit arises from $\ell_{1,q}$ being a

```
Input: Subroutine for computing $\mathrm{prox}(\boldsymbol{v},\theta,q)$; vector $\boldsymbol{v}$; scalar $\gamma > 0$
Output: $\boldsymbol{u}^* = \mathrm{argmin}_{\boldsymbol{u}} \|\boldsymbol{u}-\boldsymbol{v}\|_2$, s.t. $\|\boldsymbol{u}\|_{1,q} \le \gamma$
if $\|\boldsymbol{v}\|_{1,q} \le \gamma$ then
    return $\boldsymbol{u}^* = \boldsymbol{v}$
else
    Define $g(\theta) := -\gamma + \|\mathrm{prox}(\boldsymbol{v},\theta,q)\|_{1,q}$;
    Compute root-bracket $(\theta_{\min},\theta_{\max}) = (0, \|\boldsymbol{v}\|_{\infty,q^*})$;
    Compute root $\theta^* = \textsc{FindRoot}(g(\theta),\theta_{\min},\theta_{\max})$;
end
return $\boldsymbol{u}^* = \mathrm{prox}(\boldsymbol{v},\theta^*,q)$
```

Algorithm 1. Projection via proximity using root-finding

sum of independent terms over the G groups (recall that $\boldsymbol{v} = [\boldsymbol{v}_1,\ldots,\boldsymbol{v}_G]$). Owing to this separability, the computation (13) decomposes into G independent, proximity subtasks, one for each $\boldsymbol{v}_g$, for $1 \le g \le G$.

We reiterate that unlike [21] or other authors who solve $\ell_{1,q}$-norm penalized regressions, our setup (for general q) is harder: we solve an $\ell_{1,q}$-**constrained** regression. This requires computing $\boldsymbol{u}(\theta)$ for several values of θ.

To simplify notation, in the sequel we use $\boldsymbol{u}(\theta)$, $\boldsymbol{v}$, and $\mathrm{prox}(\boldsymbol{v},\theta,q)$ to refer to an arbitrary ***single group*** of variables, so that the proximity task at hand is

$$\boldsymbol{u}(\theta) := \mathrm{argmin}_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u}-\boldsymbol{v}\|_2^2 + \theta\|\boldsymbol{u}\|_q, \quad q \ge 1. \tag{15}$$

First, we mention existing proximity algorithms for solving (15), and then, we present some new results in Section 2.3.1.

For $q = 1$, (15) reduces to the well-known *soft-thresholding* operation [10]:

$$\boldsymbol{u}(\theta) = \mathrm{sgn}(\boldsymbol{v}) \odot \max(|\boldsymbol{v}| - \theta, 0), \tag{16}$$

where $\odot$ represents elementwise multiplication. For $q = 2$, the solution is again available in closed form (see e.g., [8, 11]), and is given by

$$\boldsymbol{u}(\theta) = \max(1 - \theta\|\boldsymbol{v}\|_2^{-1}, 0)\boldsymbol{v}. \tag{17}$$

For $q = \infty$, the solution is slightly more involved than (16) and (17), but can be obtained via the *Moreau decomposition* [8], which for $\mathrm{prox}(\boldsymbol{v},\theta,\infty)$ implies that

$$\mathrm{prox}(\boldsymbol{v},\theta,\infty) = \boldsymbol{v} - \mathrm{proj}(\boldsymbol{v},\theta,1). \tag{18}$$

The projection $\mathrm{proj}(\boldsymbol{v},\theta,1)$ is very well-studied: it is projection onto the ℓ_1-norm ball [17, 23, 25]. This choice of q arises most notably in grouped feature selection in multitask learning settings [19, 26, 28].

For $q > 1$ different from the choices above, the proximity problems are significantly harder. Fortunately, efficient proximity algorithms for ℓ_q-norms were recently developed in [20, 21]. These algorithms use nested root-finding subroutines: one for solving single variable nonlinear equations; one for performing bisection over a parameter akin to θ. But unlike the cases with $q \in \{1, 2, \infty\}$, due to the highly nonlinear derivatives, one can obtain only an approximate solution to the proximity operator.

2.3.1. Proximity Operators for $\|\cdot\|_{\infty,q^*}$

We now leverage the machinery developed in Sections 2.1 and 2.2 to obtain fast proximity operators for ℓ_{∞,q^*}-norms, almost as a byproduct. The first step is to invoke convex duality.

Proposition 1. *Let $\|\cdot\|$ be any norm, and $\|\cdot\|_*$ its corresponding dual norm; also let $\gamma > 0$. Then, the following two problems are dual to each other:*

$$\min_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u}-\boldsymbol{v}\|_2^2 + \gamma\|\boldsymbol{u}\|, \tag{19}$$

$$\max_{\boldsymbol{z}} \quad -\tfrac{1}{2}\|\boldsymbol{z}\|_2^2 + \boldsymbol{z}^T\boldsymbol{v}, \quad s.t.\ \|\boldsymbol{z}\|_* \leq \gamma. \tag{20}$$

Moreover, if $\boldsymbol{z}^*$ is the optimal dual solution, then the corresponding primal is given by $\boldsymbol{u}^* = \boldsymbol{v} - \boldsymbol{z}^*$.

Proof. Follows from [32, Theorem 31.5] (essentially Moreau's decomposition).

Proposition 1 instantly yields the following useful corollary.

Corollary 1. *A problem dual to the ℓ_{∞,q^*}-proximity operator*

$$\min_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u}-\boldsymbol{v}\|_2^2 + \gamma\|\boldsymbol{u}\|_{\infty,q},$$

is the $\ell_{1,q}$-norm projection task (q is conjugate to q^*):

$$\min_{\boldsymbol{z}} \quad \tfrac{1}{2}\|\boldsymbol{z}-\boldsymbol{v}\|_2^2, \quad s.t. \quad \|\boldsymbol{z}\|_{1,q} \leq \gamma. \tag{21}$$

Proof. Lemma 1 tells us that $\|\cdot\|_{1,q^*}$ is dual to $\|\cdot\|_{\infty,q}$. Consequently, Proposition 1 implies the result (21) (after completing squares).

Corollary 1 allows computing ℓ_{∞,q^*}-norm proximity by replacing it with the corresponding $\ell_{1,q}$-norm projection, which in turn can be solved by Algorithm 1.

3 Experimental Results

In the discussion above we presented the following two main algorithms:

(i) The SPG based method iteration (4) for solving (2);
(ii) Algorithm 1, a method for computing the $\ell_{1,q}$-norm projection (5).

As previously mentioned, for $q = 1$, the projections reduce to the classical ℓ_1-norm ball problem, while for $q = 2$, already highly efficient methods are available [5]. The case $1 < q < \infty$ seems to be largely unstudied, while for $q = \infty$, recently an efficient method was proposed [28]. Thus, we begin our numerical results by evaluating our projection algorithms for computing projections onto the $\ell_{1,\infty}$-norm ball.

Note: It is not one of the aims of this paper to compare different MTL formulations (depending on different choices of q). Our main aim is to show scalability that comes from having fast proximity operators, which we highlight by using them as subroutines in a larger problem. The subroutines themselves can be used in any proximal splitting method that solves $\ell_{1,q}$-norm constrained problems. However, a longer version of this paper will include more detailed experimental evaluation.

Table 1. Runtime and accuracy for QP and FP on a $50,000 \times 10,000$ matrix $\boldsymbol{V}$

$\frac{\gamma}{\Vert\boldsymbol{V}\Vert_{1,\infty}}$	QP_{time} (s)	$\mathbf{FP_{time}}$ **(s)**	**Speedup**	$\text{QP}_{\text{accuracy}}$	$\text{FP}_{\text{accuracy}}$	QP_{obj}	FP_{obj}
0.01	22719.76	**28.26**	**804.09**	7.76E-09	**7.28E-12**	6.8778E+03	6.8778E+03
0.05	20165.31	**24.08**	**837.38**	7.33E-09	**7.09E-11**	6.1387E+03	6.1387E+03
0.10	17064.19	**23.80**	**716.93**	5.69E-09	**5.82E-11**	5.2850E+03	5.2850E+03
0.20	11491.00	**24.74**	**464.40**	8.32E-09	**2.47E-10**	3.8133E+03	3.8133E+03
0.30	7046.42	**24.89**	**283.11**	4.98E-09	**4.44E-10**	2.6484E+03	2.6484E+03
0.40	3933.99	**29.69**	**132.52**	1.64E-08	**8.59E-10**	1.7656E+03	1.7656E+03
0.50	1982.77	**31.03**	**63.90**	9.30E-09	**5.82E-11**	1.1263E+03	1.1263E+03
0.60	905.22	**31.33**	**28.89**	1.56E-09	**7.13E-10**	6.8445E+02	6.8445E+02
0.70	380.78	**29.41**	**12.95**	**4.21E-09**	6.29E-09	3.9254E+02	3.9254E+02

3.1 Projection onto the $\ell_{1,\infty}$-Ball

For ease of comparison, we use the notation of [28], which seems to be the currently standard method. In their notation, the projection task is to solve

$$\min_{\boldsymbol{W}} \quad \tfrac{1}{2}\|\boldsymbol{W}-\boldsymbol{V}\|_{\mathrm{F}}^2, \quad \text{s.t.} \quad \sum\nolimits_{i=1}^{d} \|\boldsymbol{w}^i\|_\infty \le \gamma, \tag{22}$$

where $\boldsymbol{W}$ is a $d \times n$ matrix, and $\boldsymbol{w}^i$ denotes the ith row of $\boldsymbol{W}$.

In our comparisons below, we refer to [28]'s algorithm, which was available as C code[1], as 'QP'; our fast projection method is called 'FP'. We show numerical results for a representative large-sized problem, and to stress-test both QP and FP we show results on a large dense matrix. In particular, we use a matrix $\boldsymbol{V} \in \mathbb{R}^{50,000\times 1000}$ having entries drawn following $\mathcal{N}(0,1)$. Note that the matrix $\boldsymbol{V}$ has a total of 50 million nonzero entries.

We compute the optimal $\boldsymbol{W}^*$, as γ varies from $0.01\|\boldsymbol{V}\|_{1,\infty}$ (more sparse; difficult) to $0.7\|\boldsymbol{V}\|_{1,\infty}$ (less sparse; easy) settings. Table 1 presents the associated running times, objective function values, and accuracies (accuracy is measured by the constraint violation: $|\lambda - \|\boldsymbol{W}^*\|_{1,\infty}|$, for an estimated $\boldsymbol{W}^*$).

Minor numerical differences between both algorithms are due to unavoidable floating-point round-off errors. Also noteworthy is the fact that although QP is an "exact" method, and FP is based on root-finding, the latter ends up obtaining solutions of higher accuracy than QP. The results indicate the tremendous advantages that our method offers for large-scale data, where it vastly outperforms the competition.

3.2 Projection onto $\ell_{1,q}$-Balls

Our next experiment shows running time results for computing projection onto $\ell_{1,q}$ balls; in our experiment we selected $q \in \{1.5, 2.5, 3, 5\}$. Here too, we use matrix-based groups and solve

$$\min_{\boldsymbol{W}} \quad \|\boldsymbol{W}-\boldsymbol{V}\|_{\mathrm{F}}^2, \quad \text{s.t.} \sum\nolimits_i \|\boldsymbol{w}^i\|_q. \tag{23}$$

For each value of q, the plots in Figure 1 also show the running times requires as the parameter γ is varied. From these plots the we observe four main points: (i)

[1] *http://www.lsi.upc.edu/∼aquattoni/CodeToShare/*

the runtimes seem to be largely independent of the value of γ; (ii) for relatively small q, the projection times are approximately same; and (iii) for larger q (here $q = 5$), the projection times increase dramatically.

Moreover, from the actual running times it is apparent our projection code scales linearly with the data size. For example, the matrix corresponding to the second bar plot has 25 times more parameters than the first plot, and the runtimes reported in the second plot are approximately 25–30 times higher.

The running times in Figure 1 suggest that although the running times scale linearly, a single $\ell_{1,q}$-norm projection still takes nontrivial effort. Thus, even though our $\ell_{1,q}$-projection method is relatively fast, currently we can recommend it only for small and medium-scale regression problems.

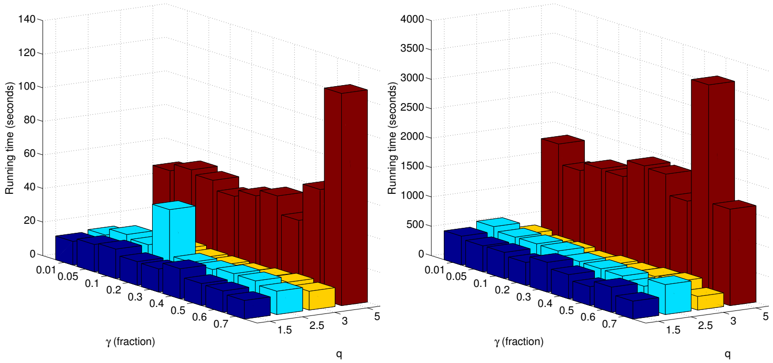


Fig. 1. Running times for $\ell_{1,q}$-norm projections as scalars q and ratios $\gamma/\|\boldsymbol{V}\|_{1,q}$ vary. The left plot is on a 1000 × 100 matrix, while the right one is on a 5000 × 500 matrix.

4 Multitask Lasso with $\ell_{1,\infty}$-Constraint

Multitask Lasso (Mtl) [19, 36] is a typical grouped feature selection problem, where important features are separated from less important ones by using information shared across multiple tasks. The feature selection is effected by a sparsity promoting mixed-norm, usually the $\ell_{1,\infty}$-norm [19].

Mtl is setup is as follows. Let $\boldsymbol{X}_j \in \mathbb{R}^{m_j \times d}$ be the data matrix for task j, where $1 \leq j \leq n$. Mtl seeks a parameter matrix $\boldsymbol{W} \in \mathbb{R}^{d \times n}$, each column of which corresponds to a task, which are regularized across the same feature by applying the mixed-norm over the rows $\boldsymbol{w}^i$ $(1 \leq i \leq d)$ of $\boldsymbol{W}$. This leads to a "grouped" feature selection, because if $\|\boldsymbol{w}^i\|_\infty = 0$, then the entire row $\boldsymbol{w}^i$ is eliminated (i.e., feature i is removed). A common formulation for Mtl is

$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_n} \quad \mathcal{L}(\boldsymbol{W}) := \sum\nolimits_{j=1}^{n} \tfrac{1}{2}\|\boldsymbol{y}_t - \boldsymbol{X}_j\boldsymbol{w}_j\|_2^2, \quad \text{s.t.} \quad \sum\nolimits_{i=1}^{d} \|\boldsymbol{w}^i\|_\infty \leq \gamma, \tag{24}$$

where the $\boldsymbol{y}_t$ are the dependent variables, and $\gamma > 0$ is a sparsity-tuning parameter. Notice that the loss-function combines the different tasks (over columns of $\boldsymbol{W}$), but the overall problem does not decompose into separable problems because the mixed-norm constrained is over the *rows* of $\boldsymbol{W}$.

Table 2. Details of simulation data used for MTL. For simplicity, all matrices X_j (for each task $1 \leq j \leq n$), were chosen to have size $m \times d$.

Name	(m, d, n)	#nonzeros	RAM required
M1	(10, 5000, 20)	10^6	7.63MB
M2	(100, 5000, 20)	10^7	76.29MB
M3	(100, 10000, 100)	10^8	0.75GB
M4	(800, 15000, 300)	$3.6 \cdot 10^9$	26.822GB

4.1 Simulation Results for MTL

In our first set of results (Tables 2, 3) we report running time comparisons between two different invocations of an SPG-based method for solving (24), once with QP as the projection method and once with FP—we call the corresponding solvers SPG$_{\text{QP}}$, and SPG$_{\text{FP}}$.

We show simulation results on small, medium, large, and very large-scale data matrices. The key aim of our experiments is to offer strong evidence showing that for data with a large number of features, or for data having large size, SPG$_{\text{FP}}$ vastly outstrips SPG$_{\text{QP}}$.

We only compare SPG based implementations because SPG offers a simple, yet highly competitive framework for solving *constrained* convex problems. The other efficient MTL algorithms that are available, e.g., [16, 21], solve the simpler, *penalized* version of the problem. Moreover, even if we were to use a method such as PQN [33] that can handle mixed-norm constraints, the final performance of potential PQN+QP or PQN+FP combinations would exhibit trends, similar to those reported in Table 3. This is so, because of the vastly different costs of projection incurred by QP and FP. We also note in passing that methods such as block-coordinate descent that are popular for several lasso-type problems, do not scale well to the large MTL problems that we consider.

The running times shown in Table 3 suggest that SPG$_{\text{FP}}$ is a valuable choice for solving large-scale MTL problems. Note that both SPG$_{\text{QP}}$ and SPG$_{\text{FP}}$ find exactly (up to roundoff error) the same solution, as both of them just perform an equal number of SPG iterations and $\ell_{1,\infty}$-projections. The difference lies in the speed of the overall execution.

4.2 MTL Results on Real-World Data

Now we show running time results comparing SPG$_{\text{QP}}$ against SPG$_{\text{FP}}$ on a real-world dataset. These results corroborate the claims of the previous section, and indicate that the powerful speedups observed on simulated data also carry over to realistic data.

Also noteworthy is the observation that the speedups attained become more significant with increasing data dimensionality. Thus, for many machine learning and statistics datasets, the acceleration offered by our fast projection algorithms can be advantageous.

Table 3. Running time (in seconds) comparisons for four different (synthetic) MTL datasets. The MTL problems were solved to an accuracy of 10^{-5}, and the total number of projections required to reach this accuracy are reported under the column '#projs'. The columns 'proj$_{\text{QP}}$' and 'proj$_{\text{FP}}$', report the total time spent by the SPG$_{\text{QP}}$ and SPG$_{\text{FP}}$ methods, respectively, for the $\ell_{1,\infty}$-projections alone. The total time taken by SPG$_{\text{QP}}$ and SPG$_{\text{FP}}$ is reported under columns with the same name. For SPG$_{\text{QP}}$ the cost of projection dominates the total runtime, while SPG$_{\text{FP}}$ this is not the case.

Dataset	#projs	proj$_{\text{QP}}$	SPG$_{\text{QP}}$	proj$_{\text{FP}}$	SPG$_{\text{FP}}$	Speedup
M1	171	822.2	826.3	26.58	**31.7**	**26.06**
M2	31	121.9	125.9	5.6	**11.4**	**11.04**
M3	29	2433.9	2474.6	31.6	**84.2**	**29.40**
M4	16	735.2	1054.2	6.0	**321.5**	**3.28**

For our experiments with real-world data, we run MTL on a subset of the CMU Newsgroups dataset[2]. This data corresponds to 5 feature selection tasks based on data taken from the following newsgroups: computer, politics, science, recreation, and religion. The feature selection tasks are spread over the matrices $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_5$, each of size 2907×53975, while the dependent variables $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_5$ correspond to class labels.

We note that for timing experiments, the exact details of the data are not that critical, except the fact that the number of features (53975) is large; indeed much larger than in our simulations. For such a large number of features, based on the results of Table 1, one can already anticipate that SPG$_{\text{FP}}$ will strongly outperform the competition.

Table 4. Running time (in seconds) comparisons for 4 different runs of MTL on the CMU newsgroups data. Here, γ-fraction (cf. Table 1) indicates sparsity-level; low γ means high sparsity, and consequently difficult optimization. The other columns are the same as in Table 3.

(γ-fraction)	#projs	proj$_{\text{QP}}$	SPG$_{\text{QP}}$	proj$_{\text{FP}}$	SPG$_{\text{FP}}$	Speedup
0.01	85	6322.2	6806.6	96.2	**507.6**	**13.41**
0.1	94	9335.2	9759.5	92.6	**649.5**	**15.03**
0.2	97	4209.8	4746.2	112.8	**554.3**	**8.56**
0.3	99	4516.6	4951.2	109.7	**514.9**	**9.62**

5 Conclusions

In this paper we took a careful look at projections onto $\ell_{1,q}$-mixed norm balls. This projection arises as a key step (particularly for $q = 2, \infty$) in groupwise feature selection problems, such as multitask lasso or group lasso. We first presented a simple spectral projected gradient (SPG)-based method for solving

[2] Original available from: *http://www.cs.cmu.edu/∼textlearning/*. We obtained the reduced, pre-processed version from the authors of [16].

convex regression problems with $\ell_{1,q}$-norm constraints. We chose SPG as it offers a simple to use method that displays strong empirical performance. This performance, though, in mixed-norm regression problems depends on the projection step being cheap. Thus, to handle projections efficiently, we presented a generic root-finding algorithm.

Our numerical results highlighted our root-finding method, specialized to the $\ell_{1,\infty}$-norm, by comparing it against the state-of-the-art method of [28]. The speedups observed were of almost three orders of magnitude. Building on these speedups, we showed how SPG combined with our projection method leads to an effective multitask lasso algorithm. On both simulated and real-world data our numerical results indicate the added efficiency afforded by methods.

At this juncture, several directions of future work are open. The most challenging amongst them is the development of a projection based method that can outperform the well-established SPG method. Additional avenues of work include extending some of our ideas to tackle more complex structured sparsity inducing norms [1, 24].

References

1. Bach, F.: Structured sparsity-inducing norms through submodular functions. In: NIPS (2010)
2. Bach, F., Jenatton, R., Mairal, J., Obozinski, G.: Convex optimization with sparsity-inducing norms. In: Sra, S., Nowozin, S., Wright, S.J. (eds.) Optimization for Machine Learning. MIT Press, Cambridge (2011)
3. Bach, F.R.: Consistency of the Group Lasso and Multiple Kernel Learning. J. Mach. Learn. Res. 9, 1179–1225 (2008)
4. Barzilai, J., Borwein, J.M.: Two-Point Step Size Gradient Methods. IMA Journal of Numerical Analysis 8(1), 141–148 (1988)
5. van den Berg, E., Schmidt, M., Friedlander, M.P., Murphy, K.: Group sparsity via linear-time projection. Tech. Rep. TR-2008-09, Univ. British Columbia (June 2008)
6. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, Belmon (1999)
7. Birgin, E.G., Martínez, J.M., Raydan, M.: Nonmonotone Spectral Projected Gradient Methods on Convex Sets. SIAM J. Opt. 10(4), 1196–1211 (2000)
8. Combettes, P.L., Pesquet, J.: Proximal Splitting Methods in Signal Processing. arXiv:0912.3522v4 (May 2010)
9. Dai, Y.H., Fletcher, R.: Projected Barzilai-Borwein Methods for Large-scale Box-constrained Quadratic Programming. Numerische Mathematik 100(1), 21–47 (2005)
10. Donoho, D.: Denoising by soft-thresholding. IEEE Tran. Inf. Theory 41(3), 613–627 (2002)
11. Duchi, J., Singer, Y.: Online and Batch Learning using Forward-Backward Splitting. JMLR (September 2009)
12. Evgeniou, T., Micchelli, C., Pontil, M.: Learning multiple tasks with kernel methods. J. Mach. Learn. Res. 6, 615–637 (2005)
13. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: KDD (2004)

14. Friedman, J., Hastie, T., Tibshirani, R.: A note on the group lasso and a sparse group lasso. arXiv:1001.0736v1 [math.ST] (January 2010)
15. Jenatton, R., Mairal, J., Obozinski, G., Bach, F.: Proximal Methods for Sparse Hierarchical Dictionary Learning. In: ICML (2010)
16. Kim, D., Sra, S., Dhillon, I.S.: A scalable trust-region algorithm with application to mixed-norm regression. In: Int. Conf. Machine Learning (ICML) (2010)
17. Kiwiel, K.: On Linear-Time Algorithms for the Continuous Quadratic Knapsack Problem. Journal of Optimization Theory and Applications 134, 549–554 (2007)
18. Kowalski, M.: Sparse regression using mixed norms. Applied and Computational Harmonic Analysis 27(3), 303–324 (2009)
19. Liu, H., Palatucci, M., Zhang, J.: Blockwise Coordinate Descent Procedures for the Multi-task Lasso, with Applications to Neural Semantic Basis Discovery. In: Int. Conf. Machine Learning (June 2009)
20. Liu, J., Ji, S., Ye, J.: SLEP: Sparse Learning with Efficient Projections. Arizona State University (2009), http://www.public.asu.edu/~jye02/Software/SLEP
21. Liu, J., Ye, J.: Efficient L1/Lq Norm Regularization. arXiv:1009.4766v1 (2010)
22. Liu, J., Ye, J.: Moreau-Yosida Regularization for Grouped Tree Structure Learning. In: NIPS (2010)
23. Liu, J., Ye, J.: Efficient Euclidean projections in linear time. In: ICML (June 2009)
24. Mairal, J., Jenatton, R., Obozinski, G., Bach, F.: Network Flow Algorithms for Structured Sparsity. In: NIPS (2010)
25. Michelot, C.: A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. J. Optim. Theory Appl. 50(1), 195–200 (1986)
26. Obonzinski, G., Taskar, B., Jordan, M.: Multi-task feature selection. Tech. rep., UC Berkeley (June 2006)
27. Patriksson, M.: A survey on a classic core problem in operations research. Tech. Rep. 2005:33, Chalmers University of Technology and Göteborg University (October 2005)
28. Quattoni, A., Carreras, X., Collins, M., Darrell, T.: An Efficient Projection for $\ell_{1,\infty}$ Regularization. In: ICML (2009)
29. Rakotomamonjy, A., Flamary, R., Gasso, G., Canu, S.: $\ell_p - \ell_q$ penalty for sparse linear and sparse multiple kernel multi-task learning. Tech. Rep. hal-00509608, version 1, INSA-Rouen (2010)
30. Rice, U.: Compressive sensing resources (October 2010), http://dsp.rice.edu/cs
31. Rish, I., Grabarnik, G.: Sparse modeling: ICML 2010 tutorial. Online (June 2010)
32. Rockafellar, R.T.: Convex Analysis. Princeton Univ. Press, Princeton (1970)
33. Schmidt, M., van den Berg, E., Friedlander, M., Murphy, K.: Optimizing Costly Functions with Simple Constraints: A Limited-Memory Projected Quasi-Newton Algorithm. In: AISTATS (2009)
34. Similä, T., Tikka, J.: Input selection and shrinkage in multiresponse linear regression. Comp. Stat. & Data Analy. 52(1), 406–422 (2007)
35. Tropp, J.A.: Algorithms for simultaneous sparse approximation, Part II: Convex relaxation. Signal Proc. 86(3), 589–602 (2006)
36. Turlach, B.A., Venables, W.N., Wright, S.J.: Simultaneous Variable Selection. Technometrics 27, 349–363 (2005)
37. Yuan, M., Lin, Y.: Model Selection and Estimation in Regression with Grouped Variables. Tech. Rep. 1095, Univ. of Wisconsin, Dept. of Stat. (2004)
38. Zhang, Y., Yeung, D.Y., Xu, Q.: Probabilistic Multi-Task Feature Selection. In: NIPS (2010)
39. Zhao, P., Rocha, G., Yu, B.: The composite absolute penalties family for grouped and hierarchical variable selection. Ann. Stat. 37(6A), 3468–3497 (2009)

Generalized Dictionary Learning for Symmetric Positive Definite Matrices with Application to Nearest Neighbor Retrieval

Suvrit Sra[1] and Anoop Cherian[2]

[1] MPI for Intelligent Systems,
72076 Tübingen, Germany
[2] University of Minnesota, Twin Cities,
Minneapolis, MN-55414, USA

Abstract. We introduce *Generalized Dictionary Learning* (GDL), a simple but practical framework for learning dictionaries over the manifold of positive definite matrices. We illustrate GDL by applying it to Nearest Neighbor (NN) retrieval, a task of fundamental importance in disciplines such as machine learning and computer vision. GDL distinguishes itself from traditional dictionary learning approaches by explicitly taking into account the manifold structure of the data. In particular, GDL allows performing "sparse coding" of positive definite matrices, which enables better NN retrieval. Experiments on several covariance matrix datasets show that GDL achieves performance rivaling state-of-the-art techniques.

1 Introduction

Recent times have seen a steep rise of data that are encoded as matrices or tensors. Such data goes beyond traditional vector based models, and offers new means of capturing intrinsic structure. The additional structure can in turn bring several benefits, such as richer representations, robustness to noise, and perhaps even improved empirical performance.

Some successful applications that depend on matrix valued data include: multi-camera tracking based on covariance matrices derived from appearance silhouettes [11,30]; medical diagnostics via diffusion tensor imaging [1,43]; computational anatomy [23]; robust face recognition [31]; and action recognition [41], among many others.

Like the works cited above, we too focus on matrix valued data, in particular on the highly important class of symmetric positive (semi)definite (SPD) matrices (e.g., covariance, correlation, kernel matrices). We deal with SPD matrices in the context of overcomplete dictionary learning, for which we present a simple but effective, new framework called *Generalized Dictionary Learning* (GDL).

Our framework extends the idea of dictionary learning over vectors [29,10] to dictionary learning over matrices. Consequently, GDL provides an approach to perform *sparse coding* for input covariance matrices. To illustrate the benefits of such sparse coding, we show an application to Nearest Neighbor (NN)-based

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 318–332, 2011.

object retrieval—a problem of central importance in machine learning and computer vision [42,31,38]. Experiments (see Section 4) reveal that GDL leads to NN performance rivaling state-of-the-art approaches.

To help place this paper in perspective, we list below its key contributions.

- **Framework.** We extend dictionary learning to matrices, and specialize this for the broadly useful class of SPD matrices.
- **Algorithm.** For GDL we present a scalable online algorithm that also allows rapid sparse coding.
- **Application.** We apply GDL to accelerate NN-based object retrieval; GDL leads to better accuracy than many of the competing methods.

1.1 Background and Related Work

We summarize below some relevant background material, which includes a brief sketch of literature directly relevant to our paper. We begin with the key objects of this paper: *covariance matrices*[1].

Typically covariance matrices encode input data as follows. For each input object, first, a set of vector-valued features is extracted. Then, the covariance matrix of these features is computed to obtain a *structured representation* for the object in question (also see Figure 1). Note that we are *not* talking about a covariance matrix across objects—rather, there is a separate covariance matrix for *each* input object.

What makes covariance matrices so special? Apart from encoding inter-feature dependencies, a key aspect of covariance matrices that has been found to be widely useful is their natural geometric property: they inhabit a Riemmanian manifold of negative curvature [22]. This geometric property is central to several algorithms that deal with covariance data while accounting for their manifold structure [2,42,31,38]. But efficiently handling this structure is nontrivial. The difficulty stems from two main reasons: (i) defining divergence, distance, or kernel functions on covariances is not easy; and (ii) even for basic computations such as distances, clustering, etc., the numerical burden is substantial.

These burdens are especially pronounced in the context of our application: nearest neighbor retrieval, where rapid and accurate processing is crucial. Let us, therefore, briefly review the state-of-the-art techniques for NN in general, while considering how they extend to SPD matrices in particular.

Nearest Neighbors. Efficient NN retrieval on covariance data is an area still in its infancy, so literature on it is scarce. Perhaps the simplest approach to NN retrieval on covariances is to use their natural Riemannian metric, namely, the geodesic distance [12]. This is defined for two covariance matrices X and Y, as

$$d_{geo} = \|\log(\lambda_{X,Y})\| \tag{1}$$

where $\lambda_{X,Y}$ is a vector of the generalized eigenvalues of X and Y. This metric, together with the Karcher mean algorithm [30] for computing the centroids of SPD

[1] We use the terms “symmetric positive definite” and “covariance” interchangeably.

matrices opens up the possibility of using a metric tree [8] data structure. While this seems an attractive option, it is seldom used in practice as the Karcher mean is an iterative and computationally intensive algorithm, making it undesirable in data intensive applications such as NN. Thus, the main line of investigation for NN has been towards developing fast hashing schemes. A possibility for hashing is in discarding the manifold structure by vectorizing the covariances through their tangent space, albeit using the log-Euclidean projection, so that vector space techniques such as locality sensitive hashing (LSH) [14,18,21] become applicable [37]. The log-Euclidean embedding was also used by [7] to develop a spectral hashing method.

On the other hand, NN for vector valued data has been a core research topic for many decades; several data structure based algorithms can be found in classic references such as [19,28]. But it is well-recognized that exact NN is computationally prohibitive for high-dimensional data, whereby in recent years increasingly approaches based on approximate methods, such as approximate nearest neighbors [3,16,15] and LSH [14,18,21] have emerged. These techniques exploit the geometric properties of the data to assign "similar" data points to the same bucket by using carefully chosen hash functions. More recently, machine learning techniques have been suggested to learn the hash functions directly from the data itself [39,42,20]. We also take a machine learning based approach, but since our underlying data space is actually a manifold, not a vector space, existing vector-based methods do not directly apply.

Learning with SPD matrices. We propose to encode SPD matrices using a weighted *sparse* combination of rank-1 positive semidefinite matrices. This idea is natural, and has been previously explored in other contexts too. For example, in [34,35] the authors investigate this idea for Mahalanobis metric learning. A paper closer in spirit to our approach is [36], where a given covariance matrix is assumed to be representable as a sparse linear combination of SPD atoms in a tensor dictionary; the learning problem, however, is formulated as a Semidefinite Program (SDP) which makes it improbable to scale to large datasets. Finally, we note that in the compressed sensing community, optimizing over matrix-valued data by minimizing the trace norm (nuclear norm) [5,26,6] is popular. But the compressed sensing setup is orthogonal to ours: there one assumes that one has a dictionary, while in our case, we seek to learn a dictionary.

2 Generalized Dictionary Learning

Traditional dictionary learning processes input vectors (signals) $s_i \in \mathbb{R}^p$, $1 \leq i \leq m$, to construct a matrix $D \in \mathbb{R}^{p \times n}$ and vectors $c_i \in \mathbb{R}^n$ (n is the number of "basis" vectors; usually, $n \gg p$), so that

$$s_i \approx Dc_i, \quad \text{and} \quad c_i \text{ is sparse}, \quad \text{for } 1 \leq i \leq m.$$

The sparsity requirement on c_i is commonly enforced using ℓ_0- or ℓ_1-norm penalties (or constraints). Since, both D and c_i are unknown, the dictionary learning problem leads to a difficult nonconvex optimization task. Nevertheless, numerically it has been a successful approach toward sparse coding [40,10].

Now, we depart from the traditional setup above: we assume that instead of vectors, we have input matrices $S_i \in \mathbb{R}^{p\times q}$, $1 \le i \le m$. Thus, instead of a matrix D, we learn a tensor $\mathcal{D}$, which we identify with a linear operator $\mathcal{D} : \mathbb{R}^{n\times r} \to \mathbb{R}^{p\times q}$. This operator maps a matrix C_i to obtain an approximate S_i; formally,

$$S_i \approx \mathcal{D}C_i, \quad \text{and} \quad C_i \text{ is sparse}, \quad \text{for } 1 \le i \le m. \tag{2}$$

Based on (2), we propose to solve *Generalized Dictionary Learning* (GDL) by casting it as the penalized optimization problem

$$\min_{C_1,\ldots,C_m,D} \quad \tfrac{1}{2}\sum\nolimits_{i=1}^{m} \|S - \mathcal{D}C_i\|_{\mathrm{F}}^2 + \sum\nolimits_{i=1}^{m} \beta_i \mathrm{sp}(C_i), \tag{3}$$

where $\beta_i > 0$ are scalars, and the function $\mathrm{sp}(C)$ enforces some notion of sparsity. Typical choices for $\mathrm{sp}(C)$ include, the cardinality function $\|C\|_0$, its convex relaxation $\|C\|_1$, the matrix rank function $\mathrm{rank}(C)$, or its convex relaxation, the trace-norm $\|C\|_{\mathrm{tr}}$.

How does (3) apply to SPD matrices? To answer this, let us restrict the input matrices S_i to $\mathscr{S}_{++}^p$, the set of $p \times p$, SPD matrices. To ensure that the approximation $\mathcal{D}C_i$ is also SPD, we must impose some structural restrictions on both the dictionary $\mathcal{D}$ and the coefficient matrix C_i. The following easily proved, classical result from matrix algebra provides the key.

Theorem 1. *If $A \succeq 0$, and B is any matrix (of suitable size), then $BAB^T \succeq 0$.*

Theorem (1) suggests that we should encode $\mathcal{D}$ by the following bilinear map

$$\mathcal{D}C := DCD^T, \quad \text{for some matrix } D, \tag{4}$$

and additionally *restrict* to $C \succeq 0$. Notice that, viewed as a linear map (4) can be written using the 'vec' operator that stacks columns of its argument as follows:

$$\mathrm{vec}(DCD^T) = (D \otimes D)\,\mathrm{vec}(C), \tag{5}$$

where the operator $\mathcal{D}$ is identified with the product $D \otimes D$. The matrix notation in (4) looks simpler. If we were to use a general matrix in (5), the storage and computational would be much higher. It can be easily seen that (5) takes $md^2 + d^2n + mn$ storage space for m covariance matrices of dimension d each, while (4) takes $md^2 + dn + mn$. Computationally, the second formulation leads to $O(d^2n)$ per iteration cost, while the first one leads to just $O(dn)$. Thus, we prefer formulation (4).

As for the coefficient matrix C, there are two fundamental choices:

1. $C = \mathrm{Diag}(c_1, \ldots, c_n)$ where $c_i \ge 0$; and
2. $C = \sum_j^k c_j c_j^T$, a general, potentially low-rank (if $k < n$) SPD matrix.

We focus on the first choice, and it is equivalent to modeling input SPD matrices as weighted sums of rank-1 matrices; specifically,

$$S \approx DCD^T = \sum\nolimits_{i=1}^{n} c_i d_i d_i^T, \quad \text{where } c_i = C_{ii}. \tag{6}$$

Although choosing a diagonal C might appear to be simple, it is quite powerful as equation (6) suggests; more importantly, this choice prevents a parameter explosion and proves crucial for GDL's computational efficiency.

2.1 Online GDL Algorithm

Now we proceed to deriving an efficient online (stochastic-gradient based) algorithm for approximately solving the GDL problem. To keep the subproblems tractable, we use the convex function $\mathrm{sp}(C) = \|C\|_1$ for enforcing sparsity. Then, using representation (6), the GDL formulation (3) becomes

$$\min_{C_1,\ldots,C_N \geq 0, D} \quad \tfrac{1}{2}\sum\nolimits_{i=1}^{m} \|S_i - DC_iD^T\|_{\mathrm{F}}^2 + \sum\nolimits_{i=1}^{m} \beta_i \|C_i\|_1, \tag{7}$$

where $\beta_j > 0$ are sparsity-tuning scalars, and $C_1, \ldots, C_m \geq 0$ are diagonal.

Like its vector space counterpart, the GDL problem (7) is also nonconvex, which makes it extremely unlikely to obtain a globally optimal solution. Fortunately, it is individually convex in D and $(C_1, \ldots, C_m)$, which suggests that a minimization strategy that alternates between optimizing over D and the matrices $(C_1, \ldots, C_m)$, could be applied. However, often the number of input data points m is very large, whereby, the alternating step over the C_i can easily become computationally prohibitive. Taking cue from the recent work in dictionary learning [10,27], we develop an online algorithm based on stochastic gradient descent. The online approach allows our GDL algorithm to easily scale to large datasets, as long as the subproblems can be solved efficiently. We now describe the key algorithmic details.

To prevent degenerate solutions, it is often useful to impose some normalization constraints on D. A practical choice is to require $\|d_j\| \leq 1$ for each column of matrix D. We denote these requirements by the feasible set $\mathscr{D}$. To run a stochastic-gradient procedure, we break up the processing into B "mini-batches." Then, we rewrite the GDL (7) over these mini-batches as

$$\min_{D \in \mathscr{D}} \quad \Phi(D) := \sum\nolimits_{b=1}^{B} \phi_b(D), \tag{8}$$

where ϕ_b denotes the objective function for batch b. Let k_b be the size of batch b $(1 \leq b \leq B)$ that contains the input matrices $\left\{S_{j(i)} | 1 \leq i \leq k_b\right\}$, where $j(i)$ denotes an appropriate index in $1, \ldots, m$. With this notation, the objective function for batch b may be written as

$$\phi_b(D) := \min_{C_{j(1)},\ldots,C_{j(k_b)} \geq 0} \quad \tfrac{1}{2}\sum\nolimits_{i=1}^{k_b} \|S_{j(i)} - DC_{j(i)}D^T\|_{\mathrm{F}}^2 + \beta_{j(i)}\|C_{j(i)}\|_1. \tag{9}$$

Our algorithm then iteratively updates the dictionary by computing

$$D_{t+1} = \Pi_{\mathscr{D}}(D_t - \eta_t \nabla_D \phi_{b(t)}(D_t)), \quad b(t) \in [1..B],\ t = 0, 1, \ldots, \tag{10}$$

where $\Pi_{\mathscr{D}}$ denotes orthogonal projection onto $\mathscr{D}$. Standard analysis (see e.g., [13]) shows that under appropriate conditions of the stepsizes η_t, the iteration above

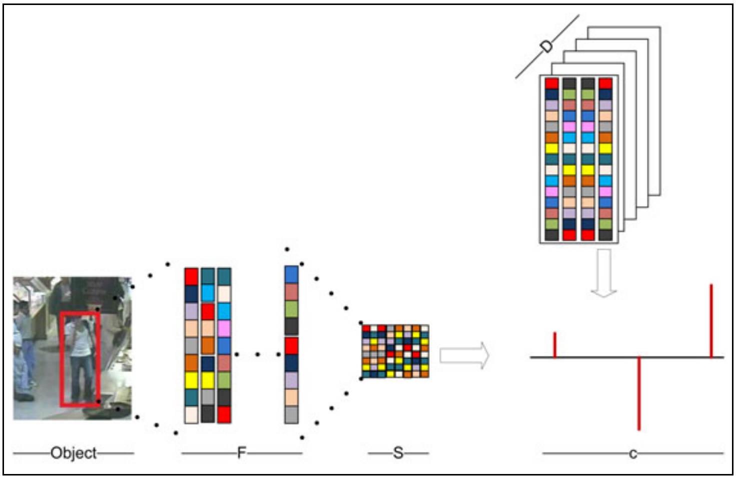


Fig. 1. Sparse coding for covariances: From the object one extracts base-features $F_1, \ldots, F_k$. These, then yield the covariance feature $S = \sum_i (F_i - \mu)(F_i - \mu)^T$ where μ is the mean feature vector, which has a sparse coding C in the dictionary $\mathcal{D}$, i.e., $S \approx \mathcal{D}C$.

converges to a stationary point of the problem. For implementing (10), note that if ϕ_b is determined by a unique solution to (9), then it can be shown that the gradient $\nabla_D \phi_{b(t)}$ is well defined. Specifically, let $(C^*_{j(1)}, \ldots, C^*_{j(k_b)})$ be the argmin of (9). Some calculus then shows that (let $b \equiv b(t)$)

$$\nabla_D \phi_b(D) = 2 \sum\nolimits_{i=1}^{k_b} \Big(DC^*_{j(i)} D^T - S_{j(i)} \Big) DC^*_{j(i)}. \tag{11}$$

2.2 Sparse Coding: Computing ϕ_b

All that remains to specify is how to compute ϕ_b, i.e., how to solve (9). First, notice that (9) is just a sum of k_b independent problems, so without loss of generality we need to consider only a subproblem of the form

$$\min_{C \geq 0} \quad f(C) := \tfrac{1}{2} \|S - DCD^T\|_{\mathrm{F}}^2 + \beta \|C\|_1, \tag{12}$$

where $\beta > 0$, and C is restricted to be a diagonal matrix. Since $C \geq 0$, problem (12) further simplifies to

$$\min_{C \geq 0} \quad f(C) := \tfrac{1}{2} \|S - DCD^T\|_{\mathrm{F}}^2 + \beta \operatorname{Tr}(C), \tag{13}$$

which is nothing but a regularized nonnegative least-squares (NNLS) problem. There exist a variety of solvers for NNLS, for example, LBFGS-B [25], SPG [4], or the very recent SBB [17]. We prefer SBB, as it is not only simple, but also exhibits strong empirical performance. Implementing SBB is simple, because it

Algorithm 1. Online Algorithm for GDL

```
Require: Input covariances $S_1, S_2 \ldots,$; stepsizes $\eta_t$
  Initialize $t \leftarrow 0$.
  while $\neg$ converged do
    Obtain next mini-batch of size $k_b$.
    for $i = 1$ to $k_b$ do
      Solve for $C_{j(i)}$ using (13).
    end for
    Update dictionary, $D_{t+1} = \Pi_{\mathscr{D}}(D_t - \eta_t \nabla_D \phi_{b(t)}(D_t))$
    $t \leftarrow t + 1$.
  end while
  return $D$.
```

only requires efficient computation of $\nabla f(C)$. Since C is diagonal, the gradient $\nabla f(C)$ is given by the diagonal matrix

$$\nabla f(C) = \mathrm{Diag}\big(D^T(DCD^T - S)D\big). \tag{14}$$

Algorithm 1 assembles all the above details into pseudocode for online GDL. Figure 1 provides a high level schema of the GDL algorithm.

3 Nearest Neighbors via GDL

Once we have a dictionary D that can sparse code an input covariance matrix S, the next step is to obtain a representation of S in terms of its sparsity. Since, we use an overcomplete dictionary of a much higher dimension than the input data, and we assume that only a few elements of the dictionary participate in the reconstruction of S, there is high probability that dissimilar input data will get unique non-zero basis combinations. In other words, suppose that we use a dictionary with n rank-1 basis matrices, and that only r of these matrices are used to reconstruct a given input covariance matrix. Then, there are $\binom{n}{r}$ unique active basis combinations possible. If n and r are chosen appropriately (by adjusting the sparsity-controlling parameters β_i), then there is a high chance that each matrix gets assigned a unique set of rank-1 matrices that encode it.

Using this observation, we hash input covariances by computing a sorted tuple representation composed of the indices (in the dictionary) of the rank-1 basis matrices involved in the reconstruction. Since each dictionary basis spans a subspace (not necessarily uniquely), the tuple may be viewed as a subspace combination corresponding to the input data. Thus, we call this representation a *Subspace Combination Tuple* (SCT), and define it formally as follows:

Definition 1. *Let $S \in \mathscr{S}^p_{++}$ be an input SPD matrix, $D \in \mathbb{R}^{p\times n}$ an overcomplete dictionary, and $\boldsymbol{u}_i$, $i \in \{1, \cdots n\}$ a unique identifier for the ith column of D. If $c = (c_1, c_2, \cdots, c_n)$ is the coefficient vector corresponding to the sparse representation of S as per (13), then a tuple $h(S) = \langle \boldsymbol{u}_i, \cdots, \boldsymbol{u}_k\rangle$ is defined as a* Subspace Combination Tuple *(SCT), if $\forall j \in \{i, \cdots,\ k\}$, $|c_j| > \epsilon$, for some $\epsilon > 0$, and $\{i, \cdots, k\}$ is a strictly increasing sequence of dictionary indices.*

In our case, we assume that the u_i's are just integers from $1, \cdots, n$ and that the hashing tuple is a set of these indices in sorted order. The threshold ϵ helps to select significant coefficients from the sparse coding, so that the chosen nonzero indices are robust to noise. This encoding of input SPD matrices as tuples opens up the possibility of using hash tables for fast locality sensitive hashing. Figure 2 illustrates this idea. Each column of the dictionary is identified by its index; so each hash-key is a set of integers encoded as a character string. To tackle collisions in the hash buckets, the colliding input matrices are organized in a linked list. If the linked list gets too long, the data within a hash bucket can be further organized using a metric tree or any other efficient data structure.

Given a query (SPD matrix), we solve (13) to first obtain its coefficient matrix, from which we obtain the SCT and query the hash table. If there are several entries in a matching bucket, we run a linear scan using the geodesic distance (1) to find the best matches (the bucket can also be organized for faster than linear scans, if desired).

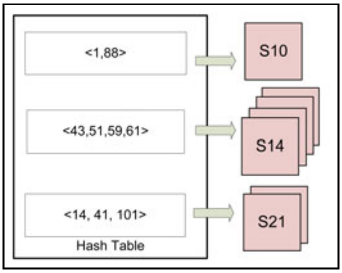


Fig. 2. An illustration of the hashing idea. The input covariance matrix is sparse coded and the nonzero active coefficients are formed into a tuple, which is then used for indexing into a hash table. To resolve hash table collisions, the colliding covariances are arranged in a suitable data structure. The covariance matrices are denoted as Si (for random values of i) in the figure.

4 Experiments and Results

We now present experimental results of applying the GDL framework to NN retrieval. Specifically, we perform extensive experiments of GDL, and compare it against the state-of-the-art NN techniques applied to NN retrieval for covariance matrices.

Since there are no publicly available datasets of covariance matrices, we had to resort to deriving them from other datasets. To this end, we selected the following types of data: (i) real-world noisy data; (ii) covariance datasets of relatively large dimensions; and (iii) a dataset with a large number of points. Our key aim in selecting such data were to highlight the applicability and relevance of our method.

For (i) we used the LabelMe object annotation dataset, for (ii) the FERET face recognition dataset, and for (iii) texture datasets. Details of each of these datasets follow.

Object dataset. We obtained the LabelMe dataset[2] of annotated images. We used approximately 10K images from this dataset, where each image contains one or more annotated objects, along with bounding box information about the location of each object in the image. We extracted these annotated blobs from each image, and created covariance matrices for these blobs. The feature vector F corresponding to each pixel in the blob of interest had the form: $F = [I_R, I_G, I_B, I_x, I_y, I_{xx}, I_{yy}]$, where the first three dimensions encode the RGB color intensities, and the last four capture the first- and second-order gradients, respectively. Thus, our covariances were 7×7, and we created a dataset containing 25K covariances from this dataset.

Face recognition. Here, we downloaded the FERET face dataset [33,32]. This dataset contains facial appearances segregated into multiple classes. Each class has different views of the face of the same person for varying poses. We selected six images from each class. Inspired by the success of covariances created from Gabor filters for face recognition [24], we applied 40 Gabor filters on each image, later combining the filters into a covariance of size 40×40. We created a covariance dataset of approximately 10K descriptors using this approach.

Texture classification. Texture is an essential cue in many data mining applications like satellite imagery, industry inspection systems, etc. Thus, we used a combination of the Brodatz dataset and the Curret dataset [9] for creating a texture covariance dataset. Brodatz contained approximately 111 texture classes, while Curret contained approimately 60 classes. To create the covariance, we used the feature vector $F = [x, y, I, I_x, I_y]$, where the first two dimensions are the relative location of the pixel with respect to the texture patch, the third dimension encodes the grayscale intensity, and the last two dimensions capture the pixel gradients. Thus, each covariance is 5×5, and we created approximately 40K such covariances.

Sample images from each of these datasets are shown in Figure 3.

4.1 Methods Compared against GDL

Log-Euclidean Embedding. The main mechanism that the state-of-the-art techniques use for NN on covariance datasets is vectorization. An input SPD matrix is projected onto its tangent plane through a log-Euclidean mapping, which results in a symmetric matrix. Since this resultant matrix is not confined to the manifold of SPD matrices, it can easily be embedded into the Euclidean space through vectorization. Then, the vectorized version can be hashed using any of the conventional hashing techniques. In our experiments, we tried two popular hashing algorithms: (1) ℓ_2-distance based LSH ($h(x) = \lfloor \frac{x.r-b}{w} \rfloor$), where x is the data vector to be hashed (L2LSH), and r and b are parameters of the

[2] http://labelme.csail.mit.edu/

Fig. 3. Sample image from LabelMe object dataset (top), FERET face appearances (mid) and Brodatz texture database (bot)

hash function; and (2) using Hamming functions (HAM) via using the binary encoding of the embedded vector.

Simple vectorization. A mere vectorization of the covariance matrix that identifies it with in Euclidean space (without projecting it using the Log-Euclidean mapping, but rather by just stacking columns), does not lead to a good hashing (VEC). However, this provides a baseline.

Kernelized LSH. A recently proposed, sophisticated technique built on LSH, is Kernelized LSH (KLSH) [20]. A major impediment in using KLSH on the space of covariances is the lack of known, effective kernel functions. We experimented with a number of potential kernels on SPD matrices (e.g., trace-kernel, log-Euclidean kernel, etc.), but found KLSH's performance to be the best when using the Riemannian kernel (which is actually a pseudo-kernel because it fails to be positive definite) generated by the Riemannian metric [12]. This kernel K has the following form: For two SPD matrices X and Y,

$$K(X, Y) := e^{-\gamma \|\log(\lambda(X,Y))\|^2}, \tag{15}$$

where $\gamma > 0$ is a "bandwidth" parameter, and $\lambda(X, Y)$ stands for the vector of generalized eigenvalues of X and Y.

4.2 Dictionary Learning

To determine the correct dictionary size for each of the datasets, we used cross validation. Assuming the dataset $S_i \in \mathscr{S}^p_{++}, i = 1, 2, \cdots, n$, we computed dictionaries of sizes $p \times kp$ for $k = 2, 3, \cdots$. For each of the dictionary, we compute the hashing accuracy on a subset of the dataset, and used the dictionary for which the best accuracy was best. This resulted in a dictionary of size 7×28 for the object dataset, 40×160 for the faces dataset, and 5×50 for the texture dataset. GDL took approximately 20 minutes for the texture dataset with 30K, 5×5 matrices, and approximately 2 hours for faces with 10K, 40×40 matrices. We ran 300 iterations of sparse coding and dictionary learning. Each of the sparse coding subproblems was seen to converge in 50–80 iterations of the SPG algorithm. For regularization, we set all β values to the same value—$\beta = 0.05$ for objects, $\beta = 0.65$ for faces, and $\beta = 0.4$ for texture, respectively. These values for β were also determined via cross-validation, while ensuring that the tuple size remained between 5–10.

4.3 Experimental Setup

GDL was implemented in Matlab; for L2LSH, VEC, and HAM we used the C-implementation from the Caltech Toolbox[3]. Since the programs have different computational baselines, we cannot compare their retrieval speed. Rather, we show in Table. 1 the average portion of each of the datasets scanned by GDL to find the nearest neighbor. The geodesic distance was used to resolve hash table collisions. As is seen from the table, the coverage (amount of database scanned) is small, which is exactly as desired.

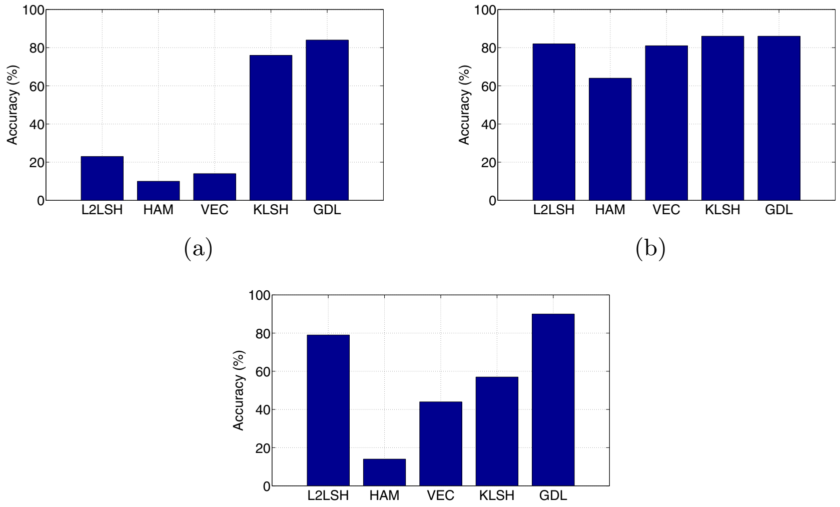
Table 1. Percentage of the database searched by GDL to find the nearest neighbor

Dataset	Objects	Faces	Texture
Avg. coverage (%)	5.11	3.54	6.26

Next, we substantiate the effectiveness of our algorithm for NN retrieval over the three datasets. For this purpose, we split each of the datasets into database and query sets (approximately 5% of the data). To compute the ground truth, we used a linear scan via the geodesic distance over the entire database for each query. Since it is hard to find exact NN, we restrict ourselves to Approximate Nearest Neighbors (ANN). Assume Q is a query point, X_{ls} is the exact NN found by a linear scan and X_{algo} is the neighbor returned by an NN algorithm. If d_{geo} defines the geodesic distance, then we classify an NN as correct if $\frac{d_{geo}(Q,X_{ls})}{d_{geo}(Q,X_{algo})} > \epsilon$. We used $\epsilon = 0.75$. Fig. 4 shows the accuracy of NN for each of the datasets, where accuracy is defined as:

$$\text{Accuracy} := \frac{\#\text{correct matches}}{\#\text{query size}}. \tag{16}$$

[3] http://www.vision.caltech.edu/malaa/software/research/image-search/



(a)

(b)

(c)

Fig. 4. Plots of the *accuracy* of NN retrieval of GDL compared to various techniques; (a) objects dataset, (b) faces dataset, and (c) texture dataset

As is evident from the plots, GDL performs well across all datasets, while the performance of the other methods vary. The vectorization approach fail on all datasets, while KLSH performed reasonably well. However, a disadvantage of KLSH compared with our method is that the former needs to compute the kernel matrix for the query point, against the *entire* dataset—this slows it down drastically. On the face dataset, all methods had high accuracy, most probably because this dataset is noise free. The other data are predominantly either outdoor images (as in LabelMe) or heavy changes in the reflectance in texture (as in the Curret dataset). For such data, adjusting the regularization parameter helps counter the effects of noise.

5 Conclusions

In this paper, we introduced a novel dictionary learning algorithm for SPD matrices which represents each input SPD matrix as a non-negative, sparse linear combination of rank-1 dictionary atoms. The key advantages of this framework are (i) the algorithm is simple and scalable, (ii) it enables fast and accurate NN on covariance datasets. We substantiated our claims by showing several experiments on real-world data. Going forward, it will be interesting to see how our framework can be extended to other matrix manifolds, beyond just SPD matrices. Another area is learning distance metric on tensors using the sparse framework, which will enable a classification or regression for SPD datasets.

References

1. Alexander, D., Pierpaoli, C., Basser, P., Gee, J.: Spatial transformations of diffusion tensor magnetic resonance images. IEEE Tran. Med. Imaging 20(11), 1131–1139 (2002)
2. Arsigny, V., Fillard, P., Pennec, X., Ayache, N.: Log-Euclidean metrics for fast and simple calculus on diffusion tensors. Magnetic Resonance in Medicine 56(2), 411–421 (2006)
3. Arya, S., Mount, D., Netanyahu, N., Silverman, R., Wu, A.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM (JACM) 45(6), 891–923 (1998)
4. Birgin, E., Martínez, J., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM Journal on Optimization 10(4), 1196–1211 (2000)
5. Cai, J., Candes, E., Shen, Z.: A singular value thresholding algorithm for matrix completion. Arxiv preprint arXiv:0810.3286 (2008)
6. Candes, E., Plan, Y.: Matrix completion with noise. Proceedings of the IEEE 98(6), 925–936 (2010)
7. Chaudhry, R., Ivanov, Y.: Fast Approximate Nearest Neighbor Methods for Non-Euclidean Manifolds with Applications to Human Activity Analysis in Videos. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6312, pp. 735–748. Springer, Heidelberg (2010)
8. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In: Proceedings of the 23rd VLDB Conference, Athens, Greece, pp. 426–435 (1997)
9. Dana, K., Van Ginneken, B., Nayar, S., Koenderink, J.: Reflectance and texture of real-world surfaces. ACM Transactions on Graphics (TOG) 18(1), 1–34 (1999)
10. Elad, M., Aharon, M.: Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. IEEE Tran. Image Processing 15(12), 3736–3745 (2006)
11. Porikli, F., Tuzel, O.: Covariance tracker. In: CVPR (2006)
12. Forstner, W., Moonen, B.: A metric for covariance matrices. Qua vadis geodesia, pp. 113–128 (1999)
13. Gaivoronski, A.A.: Convergence properties of backpropagation for neural nets via theory of stochastic gradient methods. Part 1. Optimization Methods and Software 4(2), 117–134 (1994)
14. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 518–529 (1999)
15. Indyk, P.: On approximate nearest neighbors in non-euclidean spaces. In: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, p. 148 (1998)
16. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 604–613 (1998)
17. Kim, D., Sra, S., Dhillon, I.: A non-monotonic method for large-scale non-negative least squares. Preprint on: Optimization Online (2011)
18. Kleinberg, J.: Two algorithms for nearest-neighbor search in high dimensions. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, p. 608 (1997)
19. Knuth, D.: The art of computer programming. Sorting and Searching, vol. 3. Addison-Wesley, Reading (1973)

20. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: ICCV (2009)
21. Kushilevitz, E., Ostrovsky, R., Rabani, Y.: Efficient search for approximate nearest neighbor in high dimensional spaces. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, p. 623 (1998)
22. Lang, S.: Fundamentals of differential geometry. Graduate Texts in Mathematics, vol. 191 (1999)
23. Lepore, N., Brun, C., Chou, Y., Chiang, M., Dutton, R., Hayashi, K., Luders, E., Lopez, O., Aizenstein, H., Toga, A., et al.: Generalized tensor-based morphometry of HIV/AIDS using multivariate statistics on deformation tensors. IEEE Tran. Med. Imaging 27(1), 129–141 (2007)
24. Liu, C.: Gabor-based kernel PCA with fractional power polynomial models for face recognition. IEEE PAMI 26(5), 572–581 (2004)
25. Liu, D., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical Programming 45(1), 503–528 (1989)
26. Liu, Z., Vandenberghe, L.: Interior-point method for nuclear norm approximation with application to system identification. SIAM Journal on Matrix Analysis and Applications 31(3), 1235–1256 (2009)
27. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online dictionary learning for sparse coding. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 689–696. ACM, New York (2009)
28. Mehlhorn, K.: Data structures and algorithms 3: multi-dimensional searching and computational geometry. Springer-Verlag New York, Inc., New York (1984)
29. Murray, J., Kreutz-Delgado, K.: Sparse image coding using learned overcomplete dictionaries. Machine Learning for Signal Processing, 579–588 (September 2004)
30. Tuzel, O., Porikli, F., Meer, P.: Covariance Tracking using Model Update Based on Lie Algebra. In: CVPR (2006)
31. Pang, Y., Yuan, Y., Li, X.: Gabor-based region covariance matrices for face recognition. IEEE Tran. Circuits and Sys. for Video Tech. 18(7), 989–993 (2008)
32. Phillips, P., Moon, H., Rizvi, S., Rauss, P.: The FERET evaluation methodology for face-recognition algorithms. Pattern Analysis and Machine Intelligence 22(10), 1090–1104 (2000)
33. Phillips, P., Wechsler, H., Huang, J., Rauss, P.: The FERET database and evaluation procedure for face-recognition algorithms. Image and Vision Computing 16(5), 295–306 (1998)
34. Shen, C., Welsh, A., Wang, L.: PSDBoost: Matrix-generation Linear Programming for Positive Semidefinite Matrices Learning. In: Advances Neural Information Processing Systems (2008)
35. Shen, C., Kim, J., Wang, L.: Scalable large-margin mahalanobis distance metric learning. Neural Networks 21(9), 1524–1530 (2010)
36. Sivalingam, R., Boley, D., Morellas, V., Papanikolopoulos, N.: Tensor sparse coding for region covariances. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 722–735. Springer, Heidelberg (2010)
37. Turaga, P., Chellappa, R.: Nearest-neighbor search algorithms on non-Euclidean manifolds for computer vision applications. In: Indian Conf. Comp. Vis. Graph. and Img. Proc., pp. 282–289 (2010)
38. Wang, C., Blei, D., Fei-Fei, L.: Simultaneous image classification and annotation. In: Computer Vision and Pattern Recognition (2010)
39. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. Advances in Neural Information Processing Systems 21, 1753–1760 (2009)

40. Wright, J., Ma, Y., Mairal, J., Spairo, G., Huang, T., Yan, S.: Sparse representation for computer vision and pattern recognition. In: CVPR (2009)
41. Yuan, C., Hu, W., Li, X., Maybank, S., Luo, G.: Human action recognition under log-euclidean riemannian metric. In: ACCV, pp. 343–353 (2010)
42. Zhang, H., Berg, A., Maire, M., Malik, J.: SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In: Computer Vision and Pattern Recognition, vol. 2, pp. 2126–2136. IEEE, Los Alamitos (2006)
43. Zhu, H., Zhang, H., Ibrahim, J., Peterson, B.: Statistical analysis of diffusion tensors in diffusion-weighted magnetic resonance imaging data. Journal of the American Statistical Association 102(480), 1085–1102 (2007)

Network Regression with Predictive Clustering Trees

Daniela Stojanova[1], Michelangelo Ceci[2], Annalisa Appice[2], and Sašo Džeroski[1]

[1] Jožef Stefan Institute, Department of Knowledge Technologies
Jamova cesta 39, SI-1000 Ljubljana, Slovenia
{daniela.stojanova,saso.dzeroski}@ijs.si
[2] Dipartimento di Informatica, Università degli Studi di Bari "Aldo Modo"
via Orabona 4, I-70126 Bari, Italy
{ceci,appice}@di.uniba.it

Abstract. Regression inference in network data is a challenging task in machine learning and data mining. Network data describe entities represented by nodes, which may be connected with (related to) each other by edges. Many network datasets are characterized by a form of autocorrelation where the values of the response variable at a given node depend on the values of the variables (predictor and response) at the nodes connected to the given node. This phenomenon is a direct violation of the assumption of independent (i.i.d.) observations: At the same time, it offers a unique opportunity to improve the performance of predictive models on network data, as inferences about one entity can be used to improve inferences about related entities. In this paper, we propose a data mining method that explicitly considers autocorrelation when building regression models from network data. The method is based on the concept of predictive clustering trees (PCTs), which can be used both for clustering and predictive tasks: PCTs are decision trees viewed as hierarchies of clusters and provide symbolic descriptions of the clusters. In addition, PCTs can be used for multi-objective prediction problems, including multi-target regression and multi-target classification. Empirical results on real world problems of network regression show that the proposed extension of PCTs performs better than traditional decision tree induction when autocorrelation is present in the data.

1 Introduction

Network data describe entities represented by nodes generally of the same type such as web-pages or telephone accounts, which may be connected with (related to) each other by edges which represent various explicit relations such as hyperlinks between web-pages or people calling each other. Recently, data networks such as social networks, financial transaction networks, sensor networks and communication networks have become ubiquitous in everyday life. This ubiquity of data networks motivates the recent focus of research in data mining to extend traditional inference techniques in order to learn in network data.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 333–348, 2011.

However, the extension of the traditional inference techniques opens several issues when dealing with data networks. In particular, many data networks are characterized by a form of autocorrelation where the values of the response variable at a given node depend on the values of the variables (predictor and response) at the nodes connected to the given node. Here autocorrelation is defined as the property that a value observed at a node depends on the values observed at neighboring nodes in the network. Different definitions of autocorrelation are in use depending on the field of study which is being considered and not all of them are equivalent. In statistics, autocorrelation is defined as the cross-correlation of a variable with itself at certain time lag. In spatio-temporal and time-series analysis, spatial autocorrelation has been defined as the correlation among data values, which is strictly due to the relative location proximity of the objects that the data refer to. It is justified by the Tobler's [13] first law of geography according to which "everything is related to everything else, but near things are more related than distant things" whereas in network studies the autocorrelation is defined by the homophily's principle as the tendency of nodes with similar values to be linked with each other [16]. The major difficulty due to the autocorrelation is that the independence assumptions (i.i.d.), which typically underlies machine learning methods, are no longer valid. The violation of the instance independence has been identified as the main responsible of poor performance of traditional machine learning methods [20]. To remedy the negative effects of the violation of independence assumptions, autocorrelation has to be explicitly accommodated in the learned models.

Recently, there has been a number of methods that consider the autocorrelation phenomenon and their success depends on the characteristics of the target domain. One limitation of models that represent and reason with global autocorrelation is that the methods assume the autocorrelation dependencies are stationary throughout the relational data graph [1]. Recent research has explored the use of collective inference techniques to exploit this phenomenon. These techniques achieve more accurate predictions than traditional algorithms that predict data instances individually, without regard to the relationships or statistical dependencies that are prevalent in networked data. Collective inference techniques, on the other hand, collectively predict the target values of related instances simultaneously using similarities that appear among groups of similar objects, for example tendency of friends to share political beliefs, siblings to have similar speech patterns, and linked webpages to have similar topics.

In this work, we develop an approach to modeling ('labeling') non-stationary autocorrelation in network data by using predictive clustering [3]. Predictive clustering combines elements from both prediction and clustering. As in clustering, clusters of examples that are similar to each other are identified, but a predictive model is associated to each cluster. The predictive model assigns new instances to clusters based on their description and provides a prediction for the target property. The benefit of using predictive clustering methods, as in conceptual clustering [18], is that, besides the clusters themselves, they also provide symbolic descriptions of the constructed clusters. However, in contrast to conceptual clustering, predictive clustering is a form of supervised learning.

Predictive clustering trees (PCTs) are tree structured models that generalize decision trees. Key properties of PCTs are that *i)* they can be used to predict many or all attributes of an example at once, *ii)* they can be applied to a wide range of prediction tasks (classification and regression) and *iii)* they can work with examples represented by means of a complex representation [8], which is achieved by plugging in a suitable distance metric for the task at hand, and *iv)* their tree structure permits to consider different effects of autocorrelation (non-stationariness). In the context of this paper, PCTs can be easily extended to network data in order to take (local and global) autocorrelation into account. The method extends the predictive clustering framework implemented in the CLUS system [3][1]. Given a fully described network (nodes and edges), we evaluate our models on real-world data networks (among them several geographical data networks), comparing to models that reason regardless to the global and local dependencies into the network.

The paper is organized as follows. The next section reports relevant related work. Section 3 describes the proposed approach. Section 4 describes the datasets, experimental setup and reports relevant results. Finally, in Section 5 some conclusions are drawn and some future work outlined.

2 Related Work

The motivation for this work comes from research reported in the literature for mining networked data and predictive clustering. In the following subsections, we discuss background and related work from both research lines.

2.1 Mining Network Data

In the recent years, numerous algorithms have been designed for modeling a partially labeled network and providing estimates of unknown labels associated with nodes. In general, network learning assumes that data for inference are already in the form of a network and exploit the structure of the network to allow the collective inference. Collective inference targets learning algorithms where various interrelated values are inferred simultaneously such that estimates of neighboring labels influence one another [14,10,24]. Since exact inference is known to be an NP-hard problem and there is no guarantee that data network satisfy the conditions that make exact inference tractable for collective learning, most of the research in collective learning has been devoted to the development of approximate inference algorithms.

Popular approximate inference algorithms include iterative inference, Gibbs sampling, loopy belief propagation and mean-field relaxation labeling. An outline of strengths and weakness of these algorithms is reported in [24]. In general, one of the major advantages of collective learning lies in its powerful ability to learn

[1] The CLUS system is available at http://www.cs.kuleuven.be/∼dtai/clus

various kinds of dependency structures (e.g., different degrees of correlation) [12]. However, as pointed out in [19], when the labeled data is very sparse, the performance of collective classification might be largely degraded due to the lack of sufficient neighbors. This is overcome by incorporating informative "ghost edges" into the networks to deal with sparsity issues [15,19].

Interestingly learning problems similar to the tasks addressed in network learning have been recently addressed outside the areas of network learning and graph mining. In particular, in the area of semi-supervised learning and transductive learning [25] a corpus of data without links is given to the algorithms. The basic idea is to connect data into a weighted network by adding edges (in various ways) based on the similarity between entities and to estimate a function on the graph which guarantees the consistency with the label information and the smoothness over the whole graph [26]. The constraint on smoothness implicitly assumes positive autocorrelation in the graph, that is, nearby nodes tend to share the same class labels (i.e., homophily).

Due to the recent efforts of various researchers, numerous algorithms have been designed for network learning. Anyway, at the best of our knowledge, these algorithms address the prediction problem only in the classification case. Exclusively, in an recent work of Appice et.al. [2] the authors have considered the problem of within network regression with an approach that follows the main idea of iterative inference described in [24]. The learning process resorts in the transductive setting, it is robust to sparse labeling and low label consistency and improves traditional model tree induction across a range of several geographical data networks. However, no final model is provided to the user.

2.2 Building Predictive Clustering Trees

The task of learning predictive clustering trees can be formalized in this way:
Given

- a descriptive space $\mathbf{X} = X_1, X_2, \ldots X_m$,
- a target space Y,
- a set T of examples (x_i, y_i) with $x_i \in \mathbf{X}$ and $y_i \in Y$

Find

- a set of hierarchically organized clusters defined according to $\mathbf{X} \times Y$,
- a predictive piecewise function $f : \mathbf{X} \rightarrow Y$, defined according to the hierarchically organized clusters.

The clusters to be found are defined on the basis of examples in T and represented according to both the descriptive space and the target space $\mathbf{X} \times Y$ (Figure 1(c)). This is different from what is commonly done in predictive modelling (Figure 1(a)) and classical clustering (Figure 1(b)), where only one of the the spaces is considered.

Note that this general formulation of the problem can take into account different aspects:

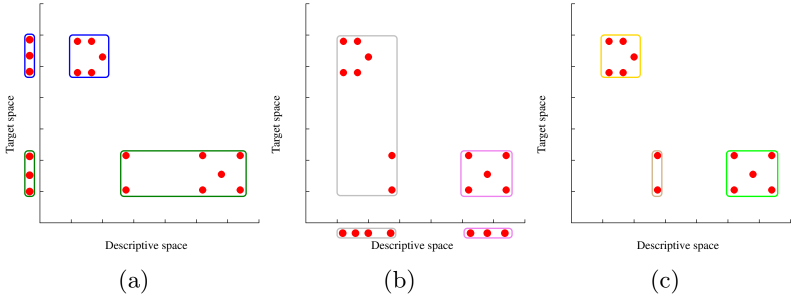


Fig. 1. Illustration of predictive clustering: (a) clustering in the target space, (b) clustering in the descriptive space, and (c) clustering in both the target and descriptive spaces. Note that the target and descriptive spaces are presented as one-dimensional axes for easier interpretation, but can be of higher dimensionality.

- multiple target attributes can be considered at the same time
- the distance function used in the clustering phase can consider the (possible) complex nature of the data
- this formulation is valid both for classification and regression problems (it depends on the nature of Y and on how the function $f(\cdot)$ is built)

In PCTs [3], a decision tree is viewed as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The construction of PCTs is not very different from that of standard decision tree learners: at each internal node t, a test has to be defined according to a given evaluation function. The main difference is that PCTs select the best test by maximizing the (inter-cluster) variance reduction, defined as $\Delta_Y(A, \mathcal{P}) = \mathit{Var}(A) - \sum_{A_k \in \mathcal{P}} \frac{|A_k|}{|A|} \mathit{Var}(A_k)$, where A represents the examples in T and $\mathcal{P}$ defines the partition $\{A_1, A_2\}$ of A.

If the variance $Var(\cdot)$ and the predictive function $f(\cdot)$ are considered as parameters, instantiated for the specific learning task at hand, it is possible to easily adapt PCTs to different domains and different tasks. To construct a regression tree, for example, the variance function returns the variance of the given instances' target values, and the predictive function is the average of target values in a cluster. Indeed, by appropriately defining the variance and predictive functions, PCTs have been used for clustering ([3]), multi-objective classification and regression ([3,7]), and time series data analysis ([8]).

In this paper, we propose to extend the problem of constructing PCTs by considering the network dimension in addition to the descriptive and target spaces, to explicitly taking autocorrelation into account.

3 Learning PCTs from Network Data

3.1 The Problem

A network is a set of entities connected by edges. Each entity is called a node of the network. A number (which is usually taken to be positive) called weight is

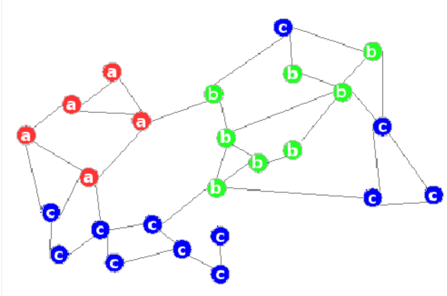


Fig. 2. Autocorrelation in network data. Different labels are given in different colors

associated with each edge. In a general formulation, a network can be represented as a (weighted) graph that is a set of nodes and a ternary relation which represent both the edges between nodes and the weight associated to each edge. Formally, a data network G is a pair $(V,\ E)$, where:
1. V is a set of nodes, and
2. E is a set of weighted edges between nodes, that is,
$G = \{\langle u, v, w\rangle | u, v \in V, w \in \Re^{+}\}$.

The network G is represented by an adjacency matrix W with entries $w_{ij} > 0$ if there is an edge between i and j, and $w_{ij} = 0$ otherwise. We impose $w_{ii} = 0$ and we define the degree of a node u_i as: $d(u_i) = \sum_j w_{ij}$. Figure 2 shows an example of a data network where different colors represent different node labels. In practice, when the original data come in the form of a network, the weight w_{ij} usually has a natural interpretation. It could be the number of hyperlinks between two web pages, or a binary value indicating whether proteins i and j interact. Many times when the weights are not readily available from the data, they are computed based on symmetric and nonnegative similarity measures. For instance, if each node is represented in the Euclidean space R_d, a popular choice is to use the Gaussian similarity measure (1)

$$w_{ij} = \exp(\| \ l_i - l_j \ \|^2 \ /2 \ b^2) \tag{1}$$

where $l_i \in R_d$ describes the location of node i and b is referred to as the bandwidth. In addition, we use a weighting function linearly dependent on the inverse Euclidean distance between objects (2) and a modified Gaussian kernel density function (3):

$$w_{ij} = (1 - \| \ l_i - l_j \ \| \ /b) \tag{2}$$

$$w_{ij} = (1 - \| \ l_i - l_j \ \|^2 \ /b^2) \tag{3}$$

which we refer to as "Euclidean" and "Modified", respectively. Whatever weighting function is selected, the estimated parameter surfaces will be, in part, functions of the definition of that weighting function.

In this work, each node of the network is associated with a data observation $(x, y) \in \mathbf{X} \times Y$. $\mathbf{X}$ is a feature space spanned by m predictor variables X_i with $i = 1, \dots, m$, while $\mathbf{Y}$ is the possibly unknown response variables with a range in $\Re$.

In order to formalize the learning task we are referring to, we need to define the network arrangement of the data with the goal of explicitly taking autocorrelation into account. For this purpose, in addition to the descriptive space X and the target space Y, it is necessary to add information on the connectedness of the data within the network (e.g., the links between the objects involved in the analysis or the pairwise distances between them).

The problem of network regression is formulated as follows.
Given:
1. a set of labeled nodes $L = \{u | u \in \mathbf{X} \times \mathbf{Y}\}$
2. a set of unlabeled nodes $U = \{u | u \in \mathbf{X}\}$
3. a neighborhood function $\eta_k : V \longmapsto (\mathbf{V} \times \Re^{+})^k$ such that:
$\eta_k(u) = \{(v_1, w_1), ..., (v_k, w_k)\}$ with $(u, v_i, w_i) \in A, i = 1...k$
Find an estimate for the unknown value of the response variables $\mathbf{Y}$ for each node $u \in U$, such that it is as accurate as possible.

3.2 Measures of Network Autocorrelation

The original CLUS algorithm uses variance reduction as an evaluation measure for the tests used in the internal nodes in the tree. However, in order to take the autocorrelation into account when partitioning the descriptive space, a different measure is necessary. Therefore, 3 autocorrelation indexes are introduced below.

In spatial data analysis, several spatial autocorrelation statistics have been defined. The most common one is the Global Moran's I [13]. This requires a spatial weight matrix that reflects the intensity of the spatial relationship between observations in a neighborhood.

The Global Moran's I is defined as

$$I_Y = \frac{N \sum_i \sum_j w_{ij}(Y_i - \overline{Y})(Y_j - \overline{Y})}{\sum_i \sum_j w_{ij} \sum_i (Y_i - \overline{Y})^2} \tag{4}$$

where N is the number of spatial objects indexed by i and j; Y_i and Y_j are the values of the variable Y for the nodes u_i and u_j, respectively; Y is the variable of interest; $\overline{Y}$ is the overall mean of Y; and $w_{ij}, i, j = 1, \dots, N$ are the values of a N x N matrix of spatial weights. Values that are more positive than expected indicate positive autocorrelation, while more negative values indicate negative autocorrelation. Values generally range from -1 to +1 and 0 indicates a random distribution of the data.

Connectivity (Randic) Index (CI) is the index of connectivity of a network (or graph) [23]. For a given network G, CI is defined by (5)

$$CI_Y(G) = \sum_{u,v \in V(G)} \frac{1}{\sqrt{d(u)d(v)}} \tag{5}$$

Algorithm 1. Top-down induction of NetworkPCTs

```
1: procedure NetworkPCT($A$) returns tree
2: if stop(A) then
3:    return leaf(Prototype($A$))
4: else
5:    $(c^*, h^*, \mathcal{P}^*) = (null, 0, \emptyset)$
6:    for each possible test $c$ do
7:       $\mathcal{P}$ = partition induced by $c$ on $A$
8:       $h = \frac{\alpha}{|Y|} \sum_{\mathbf{Y}} \Delta_Y(A, \mathcal{P}) + \frac{(1-\alpha)}{|Y|} \sum_{\mathbf{Y}} S_Y(A, \mathcal{P})$
9:       if $(h > h^*)$ then
10:         $(c^*, h^*, \mathcal{P}^*) = (c, h, \mathcal{P})$
11:      end if
12:   end for
13:   for each $A_k \in \mathcal{P}^*$ do
14:      $tree_k$ = NetworkPCT($A_k$)
15:   end for
16:   return node($c^*$, $\bigcup_k \{tree_k\}$)
17: end if
```

where $d(u)$ and $d(v)$ represent the degree of nodes u and v respectively, and $V(G)$ represents the node set. This index gives the connectedness (or branching) of a network G and can be used to compare the connectivity among networks. It is typically used in chemistry, since it can be well correlated with a variety of physico-chemical properties of alkanes, such as boiling points, surface area and solubility in water.

Relational autocorrelation measures the strength of statistical dependencies between values of a single attribute Y on related (linked) instances [1]. Any traditional measure of association, such as the χ_2 statistics or information gain, can be used to assess the association between these values of Y. For example, given a set of related pairs P_R, we can measure the autocorrelation of a continuous variable Y as the correlation between related Y_i and Y_j :

$$P_Y = \frac{\sum_{ij \ s.t. \ (u_i, u_j) \in P_R} (Y_i - \overline{Y})(Y_j - \overline{Y})}{\sum_{ij \ s.t. \ (u_i, u_j) \in P_R} (Y_i - \overline{Y})^2} \tag{6}$$

3.3 PCTs for Network Regression

The Algorithm. We can now proceed to describe the top-down induction algorithm for building PCTs from network data (Algorithm 1). It is a recursive method which takes as input a set of training instances A and partitions the descriptive space until a stopping criterion is satisfied (Algorithm 1 line 2).

The main loop (Algorithm 1, lines 6-11) searches for the best attribute-value test c^* that can be associated to a node t. It associates the best test c^* to the internal node t and calls itself recursively to construct a subtree for each subset (cluster) in the partition P^* induced by c^* on the training instances.

Possible tests are of the form $X \leq \beta$ for continuous attributes, and $X \in \{x_{i_1}, x_{i_2}, \ldots, x_{i_o}\}$ (where $\{x_{i_1}, x_{i_2}, \ldots, x_{i_o}\}$ is a subset of the domain D_X of X) for discrete attributes. For continuous attributes, possible values of β are found by sorting the distinct values of X in the training set associated to t, then considering a threshold between each pair of adjacent values. Therefore, if the cases in t have k distinct values for X, at most $k - 1$ thresholds are considered. When selecting a subset of values for a discrete attribute, we rely on a non-optimal greedy strategy [17]. It starts with an empty set $Left_t = \oslash$ and a full set $Right_t = D_X$, where D_X is the domain of X. It moves one element from $Right_t$ to $Left_t$ such that the move results in increased variance reduction. This differs from the classical solution [4], where some ordering on the possible values of D_X is defined apriori, according to the data distribution. However, the classical solution cannot deal with multi-objective predictive tasks as PCTs can.

The algorithm evaluates the best split according to the formula reported in Algorithm 1, line 8. This formula is a linear combination of the variance reduction and the statistic $S_Y(A, \mathcal{P})$, computed only on labeled examples. According to the above discussion for the selection of an appropriate evaluation measure for the tests, $S_Y(A, \mathcal{P})$ can be defined in terms of the three indexes we introduce (Moran's I, CI and and P). However, since they all range in different intervals, it is necessary to appropriately scale them. Since the variance reduction is non-negative, we decided to scale them in the interval [0,2], where 2 means high positive autocorrelation and 0 means high negative autocorrelation. For example, for Moran's I, $S_Y(A, \mathcal{P})$ is:

$$S_Y(A, \mathcal{P}) = \sum_{A_k \in \mathcal{P}} \frac{|A_k|}{|A|} \widehat{I_Y}(A_k)$$

where $\widehat{I_Y}(A_k)$ is the scaled Moran's I computed on A_k.

Moreover, in order to guarantee a fair combination of the variance reduction and the statistic $S_Y(A, \mathcal{P})$, we also need to scale the variance reduction in the interval [0,2]. For that purpose, we use a common scaling function:

$$\Delta_Y(A, \mathcal{P}) = 2\frac{\Delta_Y(A, \mathcal{P}) - \Delta min}{\Delta max - \Delta min} \tag{7}$$

where Δmax and Δmin are the maximum and the minimum values of $\Delta_Y(A, \mathcal{P})$ for a particular split.

The search stops when the number of the examples in a leaf is less than $\sqrt{N}$, which is considered a good locality threshold that does not permit to lose too much in accuracy also for rule based classifiers [11]. When the stopping criterion is satisfied, the algorithm creates a leaf and labels it with a predictive function (in this case, the average) defined for the instances falling in that leaf. When predicting multiple variables, the predictive function is an aggregation function (in this case, the average) over tuples of target values. Each target variable contributes equally to the overall h value (Algorithm 1, line 8).

Estimating the Bandwidth. The choice of the bandwidth (denoted by b in (1)) is perhaps the most critical decision to be taken in the modeling process. This parameter controls the degree of smoothing. A small bandwidth results in very rapid distance decay, whereas a larger value will result in a smoother weighting scheme. At the same time, this parameter influences the level of autocorrelation.

The bandwidth may be defined manually or by using some form of adaptive method, such as cross validation and the corrected Akaike Information Criterion (AIC), as used in GWR [9]. A wrapper solution would significantly increase (by a logarithmic factor, in the worst case) the NCLUS complexity. In this study, for the bandwidth estimation we minimize the leave-one-out cross validated - Root Mean Square Error (RMSE). Minimization is performed by means of the Golden section search [5] that aims, in this case, at binary recursively partitioning of the bandwidth domain. Partitions are not uniform in width, but maintain the *golden ratio* $\gamma = \frac{1+\sqrt{5}}{2}$. For each couple of bandwidth values, b_1 and b_2 (at the first iteration, they are initialized as minimum and maximum bandwidth, respectively), the algorithm identifies a point b_3 between them, according to the golden ratio and computes the cross validated - RMSE for that point ($RMSE_{b_3}$). The algorithm than identifies the only parabola with a vertical axis that intersects the points $\{(b_1, RMSE_{b_1}),\ (b_3,\ RMSE_{b_3}),\ (b_2, RMSE_{b_2})\}$. On the basis of the position of the minimum of this parabola, the system decides whether to consider $(b_1,\ b_3)$ or $(b_3,\ b_2)$ as the next couple of bandwidth values. The search stops when there is no cross validated - RMSE reduction. In the algorithm, RMSE is computed by fitting a weighted linear model for the example left out. A wrapper solution would significantly increase (by a logarithmic factor, in the worst case) the NCLUS complexity. While we optimize b only for (1), additional experiments have shown only minor differences among for (2) and (3).

Time Complexity. The computational complexity of the algorithm depends on the computational complexity of adding a splitting node t to the tree, which in fact depends on the complexity of selecting a splitting test for t. A splitting test can be either continuous or discrete. In the former case, a threshold a has to be selected for a continuous variable. Let N be the number of examples in the training set, then the number of distinct thresholds can be N-1 at worst. They can be determined after sorting the set of distinct values. If m is the number of descriptive variables, the determination of all possible thresholds has a complexity $O(m * N * logN)$ when an optimal algorithm is used for sorting.

For each of the possible thresholds, the system has to compute the measure used of the evaluation of a single split. This computation has, in principle, time-complexity $O(N^2)$; however, it is not necessary to recompute it at each splitting evaluation since partial sums can be incrementally updated depending on the examples that are moved from the right to the left branch. This optimization makes the complexity of the evaluation of a single split $O(N)$. This means that the worst case complexity of adding a splitting node on a continuous attribute is $O(m{*}(NlogN{+}N))$, that is $O(m{*}NlogN)$. Similarly, for a discrete splitting test, the worst case complexity is $O(m * k * N)$, where k is the maximum number of distinct values of a discrete variable ($k \leq N$). Therefore, finding the best splitting node (either continuous or discrete) has a complexity of $O(m * NlogN)$. For the induction of a complete clustering tree, this complexity, in the worst case, is $O(z * m * NlogN)$, where z is the number of internal nodes in the tree.

4 Empirical Evaluation

Before we proceed to presenting empirical results, we provide a description of the used datasets and experimental settings.

4.1 Datasets

In this experimental evaluation, we use six network data obtained from spatial datasets. They are described in the following.

NWE (North-West England) contains census data collected in the European project SPIN!. The data concerns North West England, an area that is decomposed into censual sections or wards for a total of 1011 wards. Census data provided by the 1998 Census is available at ward level. We consider percentage of mortality (target variable) and measures of deprivation level in the ward according to index scores such as the Jarman Underprivileged Area Score, Townsend score, Carstairs score and the Department of the Environments Index, as well as the coordinates of the ward centroid.

The datasets SIGMEA_MS and SIGMEA_MF (MS and MF) [7] are derived from one multi-objective dataset containing measurements of pollen dispersal (crossover) rates from two lines of plants (target variables), that is, the transgenic male-fertile (MF) and the non-transgenic male-sterile (MS) line of oilseed rape. The predictor variables are the coordinates of the sampling point, the cardinal direction and distance of the sampling point from the center of the donor field, the visual angle between the sampling plot and the donor field, and the shortest distance between the plot and the nearest edge of the donor field.

The FOIXA dataset contains measurements of the contamination rate at sampling points located within a conventional field that comes from the surrounding genetically modified (GM) fields within a 400 ha large maize production area in the Foixa region in Spain. This includes the coordinates of sampling points, the number of GM fields, the size of the surrounding GM fields, the ratio of the size of the surrounding GM field and the size of conventional field, the average distance between conventional and GM fields.

The GASD (USA Geographical Analysis Spatial Dataset) [22] contains 3106 observations on USA county votes cast in 1980 presidential election. Specifically, it contains the number of votes cast in the 1980 presidential election per county (target attribute), the coordinates, the number of owner-occupied housing units, the aggregate income and the population in each county over 18 years of age.

The Forest Fires (FF) dataset [6] is public available for research at UCI Machine Learning Repository[2]. It collects 512 forest fire observations from the Montesinho park in Portugal, including the coordinates and the burned area of the

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

forest (response variable), the Fine Fuel Moisture Code (FFMC), the Duff Moisture Code (DMC), the Drought Code (DC), the Initial Spread Index (ISI), the temperature, the relative humidity and the wind speed.

4.2 Experimental Setup

Each geo-referenced dataset is mapped into a data network $G = (V, E)$ that includes a node $u \in V$ and associates it with for each observation $(x_1, ..., x_n, y)$. Let u and v be two distinct nodes in V, there is an edge from u to v labeled with w in E (i.e., $(u, v, w) \in G$) iff v is within a bandwidth b. At the same time b is used in the weighting functions for the measures of network autocorrelation that use weights. If u and v are associated with observations taken at the same geographical site, the weighting of observations collected at this site would be unity. The weighting of other observations will decrease according to a Gaussian curve as the Euclidean distance between u and v increases. For each data network, experiments are performed in order to show the impact of the different weighting schemes (weights are defined according to equations (1), (2) and (3)).

In this paper, we consider the distances between objects based on the distances over the descriptive attributes, the spatial attributes and both of them together. Moreover, several data networks are constructed from the same dataset by varying the bandwidth b, i.e., using 1%, 5%, 10%, 20% from the maximum distance and an automatically estimated bandwidth. The bandwidth b is set using training data only.

Errors are estimated by 10 fold cross-validation. The predictive performance of the proposed system NCLUS is compared with that of the CLUS algorithm, as well as to a modification of CLUS that considers the coordinates as target variables, along with the actual response variables, for the computation of the evaluation measure (henceforth CLUS*). The latter introduces the network arrangement into CLUS without modifying the algorithm itself. We also compare with the Iterative Transductive Learning (ITL) algorithm of Appice et.al. [2] that deals with network regression tasks and also considers autocorrelation.

To evaluate the effect of different definitions of distances between objects in the network, we use the non-parametric Wilcoxon two-sample paired signed rank test [21]. To perform the test, we assume that the experimental results of the two methods compared are independent pairs $\{(q_1, r_1), (q_2, r_2), \ldots, (q_n, r_n)\}$ of sample data. We then rank the absolute value of the differences $q_i - r_i$. The Wilcoxon test statistics WT^+ and WT^- are the sum of the ranks from the positive and negative differences, respectively. We test the null hypothesis H_0: "no difference in distributions" against the two-sided alternative H_1: "there is a difference in distributions". Intuitively, when $WT^+ \gg WT^-$ and viceversa, H_0 is rejected. Whether WT^+ should be considered "much greater than" WT^- depends on the considered significance level. The basic assumption of the statistical test is that the two populations have the same continuous distribution. Since, in our experiments, q_i and r_i are average MSE, $WT^+ \gg WT^-$ implies that the second method (R) is better than the first (Q). In all experiments reported in this empirical study, the significance level used in the test is set at 0.05.

Table 1. The effect of different definitions of distances between objects in the network

Dataset	Desc.+Spatial vs. Desc.				Desc.+Spatial vs. Spatial				Desc. vs. Spatial			
	++	+	−	−−	++	+	−	−−	++	+	−	−−
NWE	3	4	3	0	2	5	5	0	0	6	6	0
MS	0	0	11	0	0	5	7	0	0	10	2	0
MF	0	6	6	0	0	12	0	0	0	6	5	0
FOIXA	0	4	4	4	1	4	4	3	3	4	3	2
GASD	11	1	0	0	0	12	0	0	0	0	12	0
FF	0	0	0	0	0	0	0	0	0	0	0	0
Total.	14	15	24	4	3	38	16	3	3	26	28	2

4.3 Results and Discussion

Table 1 shows the summary of the different definition of the distances between the objects in the network. Here, we consider three different ways of computing the distances (distance over the descriptive attributes, the spatial attributes and both of them together), all based on the computation of the Euclidean distance. For each of these, we construct the autocorrelation indices (CI, Global Moran's I), two ways of combining the variance reduction and autocorrelation ($\alpha = 0$ and $\alpha = 0.5$) and three different weighting functions for the neighborhood, given by equations (1), (2) and (3). This gives a total of 12 options: The Wilcoxon test is calculated for each of these and the summaries are given in Table 1. The results are presented in terms of the counts of the better (denoted with +) and significantly better (denoted with ++) results at significance level 0.05 for each dataset and the average counts over all datasets when calculating the distance between objects based on the distances over the descriptive attributes, the spatial attributes and both of them together. Although the results very much depend on the datasets characteristics (e.g., NWE, MF and GASD benefit from the inclusion of non-spatial attributes), the distance should be calculated over all (descriptive attributes and spatial) attributes.

The extension that we propose in this paper, introduces several parameters that can be changed by the user within the modeling process. This opens several dimensions through which one can compare the predictive performance of the proposed method, i.e., one can change the the bandwidth, weighting functions, the evaluation measure and the level of consideration of the autocorrelation.

Comparing the results obtained at different bandwidths (1%, 5%, 10%, 20%) enables us to see the influence of the neighborhood similarity on the accuracy of the results. However, in practice, this parameter is very much dependent from the specific data network. Therefore, manual selection of the bandwidth does not lead to a general conclusion for all data networks. Moreover, analysis with different bandwidths (done outside this paper due to the space limitation) confirms that an automatic estimation of the bandwidth as the one explained in 3.3, in most cases, improves the predictive power of the models obtained with a manual selection of the bandwidth. Therefore, we use the estimated bandwidth for the evaluation of the obtained results. Table 2 shows the effect of the weight-

Table 2. Average of the MSE values of NCLUS, CLUS, CLUS* and ITL. NCLUS is run with different weighting functions, evaluation measures and an automatically estimated bandwidth. For each dataset, the best results are given in bold.

Dataset	est	NCLUS CI						NCLUS P	
	b	α=0			α=0.5			α=0	α=0.5
	(%)	Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.		
NWE	7.67	0.0023	0.0023	0.0023	0.0024	**0.0022**	0.0024	0.0026	0.0024
MS	4.8	7.1220	6.1312	7.1220	6.8863	6.8863	6.8863	6.2860	7.1380
MF	9.14	2.4718	3.2346	2.4718	2.4718	2.4718	2.4718	2.4981	2.5133
FOIXA	64.62	1.0672	0.9220	1.0672	**0.7666**	0.8011	0.8011	0.9687	0.7313
GASD	2.5	0.1800	0.1808	0.1800	0.1770	0.1663	0.1780	0.1762	0.1734
FF	100	**47.224**	**47.224**	**47.224**	**47.224**	**47.224**	**47.224**	47.385	47.385

Dataset	NCLUS Global Moran						CLUS	CLUS*	ITL
	α=0			α=0.5					
	Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.			
NWE	0.0024	0.0024	0.0023	0.0023	0.0023	0.0024	0.0025	0.0025	0.0025
MS	7.1844	6.1311	7.3152	6.7851	6.9259	**5.0951**	5.9114	6.6845	5.8532
MF	2.4718	3.0922	2.4718	2.4718	2.4718	4.2877	**2.3532**	2.5390	2.4085
FOIXA	0.8231	0.8240	0.7751	0.8445	1.0201	0.7718	0.8920	0.8710	
GASD	0.1790	0.1688	0.1719	0.1695	0.1688	0.1688	0.1590	0.1590	**0.1316**
FF	**47.224**	**47.224**	**47.224**	**47.224**	**47.224**	**47.224**	47.950	47.998	64.731

ing function and its contribution within the splitting criterion. The results are presented in terms of the average Mean Square Error (MSE) for each evaluation measure, the best results are given in bold. The analysis of the results reveals that the best results are obtained by combining the Euclidian weighting function with the Global Moran spatial statistic and Gaussian weighting function with the Connectivity Index. Note that for this comparison we set $\alpha = 0$.

The selection of the user-defined parameter α is a very important step, influencing the learning process. The simplest solution is to set this parameter to 0 (consider only the network arrangement) or 1 (consider only the variance reduction for regression, as in the original CLUS algorithm). Any other solution will combine the effects, allowing both criterion to influence the split selection. Table 2 also presents the predictive performance (in terms of average MSE) of the proposed algorithm, obtained by varying the parameter α in $\{0,\ 0.5,\ 1\}$. The best results are obtained for α=0.5 for datasets where the effect of the autocorrelation is not limited to small neighborhoods. Overall, there is a gain in performance due to the used linear combination.

In Table 2, we can see that Connectivity Index CI and Global Moran I show the best measures for network autocorrelation. Moreover, we can also compare NCLUS with the original CLUS and the CLUS* version. The results show that NCLUS outperforms CLUS, CLUS* and ITL when the effect of autocorrelation are relatively high (high values of estimated bandwidth).

5 Conclusions

In this paper, we propose a data mining method that explicitly considers autocorrelation when building regression models from network data. The resulting models adapt to local properties of the data, providing, at the same time, smoothed predictions. The novelty of our approach is that, due to the generality of PCTs, it can work for different predictive modeling tasks, including regression and multi-objective regression, as well as some clustering tasks. We use well known measures of (spatial and relational) autocorrelation, since we deal with a range of several geographical data networks. The heuristic we use in the construction of PCTs is a weighted combination of variance reduction (related to predictive performance) and the autocorrelation of the response variable(s). It can also consider different sizes of neighborhoods (bandwidth) and different weighting schemes (degrees of smoothing) when calculating the autocorrelation. We identify suitable combinations of autocorrelation metrics and weighting schemes and automatically determine the appropriate bandwidth.

We evaluate our approach on six sets of geographical data. Empirical results on real world problems of network regression show that the proposed extension of PCTs performs better than both PCTs that capture local regularities but do not take into account autocorrelation and iterative transductive learner with co-training that takes into account autocorrelation. Future work will further exploit the network structure and study additional evaluation measures.

Acknowledgment. This work is in partial fulfillment of the research objectives of the project "EMP3: Efficiency Monitoring of Photovoltaic Power Plants" funded by Fondazione Cassa di Risparmio di Puglia. Džeroski is supported by the Slovenian Research Agency (grants P2-0103, J2-0734, and J2-2285), the European Commission (grant HEALTH-F4-2008-223451), the Centre of Excellence for Integrated Approaches in Chemistry and Biology of Proteins (operation no. OP13.1.1.2.02.0005) and the Jožef Stefan International Postgraduate School. Stojanova is supported by the Jožef Stefan Institute and the grant J2-2285.

References

1. Angin, P., Neville, J.: A shrinkage approach for modeling non-stationary relational autocorrelation. In: Proc. 8th IEEE Intl. Conf. on Data Mining, pp. 707–712 (2008)
2. Appice, A., Ceci, M., Malerba, D.: An iterative learning algorithm for within-network regression in the transductive setting. In: Discovery Science, pp. 36–50 (2009)
3. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: Proc. 15th Intl. Conf. on Machine Learning, pp. 55–63 (1998)
4. Breiman, L., Friedman, J., Olshen, R., Stone, J.: Classification and Regression trees. Wadsworth & Brooks, Belmont (1984)
5. Brent, R.: Algorithms for Minimization without Derivatives. Prentice-Hall, Englewood Cliffs (1973)
6. Cortez, P., Morais, A.: A Data Mining Approach to Predict Forest Fires using Meteorological Data. In: Proc. 13th Portuguese Conf. Artificial Intelligence, New Trends in Artificial Intelligence, pp. 512–523 (2007)

7. Demšar, D., Debeljak, M., Lavigne, C., Džeroski S.: Modelling pollen dispersal of genetically modified oilseed rape within the field. In: Abstracts of the 90th ESA Annual Meeting, p. 152. The Ecological Society of America (2005)
8. Džeroski, S., Gjorgjioski, V., Slavkov, I., Struyf, J.: Analysis of time series data with predictive clustering trees. In: Proc. 5th Intl. Wshp. on Knowledge Discovery in Inductive Databases, pp. 63–80. Springer, Heidelberg (2007)
9. Fotheringham, A.S., Brunsdon, C., Charlton, M.: Geographically Weighted Regression: The Analysis of Spatially Varying Relationships. Wiley, Chichester (2002)
10. Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C.: Using ghost edges for classification in sparsely labeled networks. In: Proc. 14th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining, pp. 256–264 (2008)
11. Góra, G., Wojna, A.: RIONA: A classifier combining rule induction and k-NN method with automated selection of optimal neighbourhood. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, pp. 111–123. Springer, Heidelberg (2002)
12. Jensen, D., Neville, J., Gallagher, B.: Why collective inference improves relational classification. In: Proc. 10th Intl. Conf. on Knowledge Discovery and Data Mining, pp. 593–598 (2004)
13. Legendre, P.: Spatial autocorrelation: Trouble or new paradigm? Ecology 74(6), 1659–1673 (1993)
14. Macskassy, S., Provost, F.: Classification in networked data: a toolkit and a univariate case study. Machine Learning 8, 935–983 (2007)
15. Macskassy, S.A.: Improving learning in networked data by combining explicit and mined links. In: Proc. 22th Intl. Conf. on Artificial Intelligence, pp. 590–595 (2007)
16. McPherson, M., Smith-Lovin, L., Cook, J.: Birds of a feather: Homophily in social networks. Annual Review of Sociology 27, 415–444 (2001)
17. Mehta, M., Agrawal, R., Rissanen, J.: Sliq: A fast scalable classifier for data mining. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 18–32. Springer, Heidelberg (1996)
18. Michalski, R.S., Stepp, R.: Machine Learning: An Artificial Intelligence Approach. In: Learning from Observation: Conceptual Clustering, Tioga, pp. 331–363 (2003)
19. Neville, J., Jensen, D.: Relational dependency networks. Journal of Machine Learning Research 8, 653–692 (2007)
20. Neville, J., Simsek, O., Jensen, D.: Autocorrelation and relational learning: Challenges and opportunities. In: Wshp. Statistical Relational Learning (2004)
21. Orkin, M., Drogin, R.: Vital Statistics. McGraw Hill, New York (1990)
22. Pace, P., Barry, R.: Quick computation of regression with a spatially autoregressive dependent variable. Geographical Analysis 29(3), 232–247 (1997)
23. Randic, M.: On characterization of molecular attributes. Journal of American Chemical Society (1975)
24. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 29(3), 93–106 (2008)
25. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
26. Zhu, X., Ghahramani, Z., Lafferty, J.D.: Semi-supervised learning using gaussian fields and harmonic functions. In: Proc. 20th Intl. Conf. on Machine Learning, pp. 912–919 (2003)

Learning from Label Proportions by Optimizing Cluster Model Selection

Marco Stolpe and Katharina Morik

Technical University of Dortmund, Artificial Intelligence Group
Baroper Strasse 301, 44227 Dortmund, Germany
{marco.stolpe,katharina.morik}@tu-dortmund.de
http://www-ai.cs.tu-dortmund.de/

Abstract. In a supervised learning scenario, we learn a mapping from input to output values, based on labeled examples. Can we learn such a mapping also from groups of unlabeled observations, only knowing, for each group, the proportion of observations with a particular label? Solutions have real world applications. Here, we consider groups of steel sticks as samples in quality control. Since the steel sticks cannot be marked individually, for each group of sticks it is only known how many sticks of high (low) quality it contains. We want to predict the achieved quality for each stick **before** it reaches the final production station and quality control, in order to save resources. We define the problem of learning from label proportions and present a solution based on clustering. Our method empirically shows a better prediction performance than recent approaches based on probabilistic SVMs, Kernel k-Means or conditional exponential models.

1 Introduction

Consider a steel factory where charges of steel sticks are processed sequentially at several production stations. The quality of sticks is assessed at the end of the process. For each stick though, we are given sensor measurements and other parameters during its being processed. Based on this information, we want to predict the quality of individual sticks as early as possible, *before* they reach the final production station and quality control. This saves resources, because sticks that can no longer reach the desired quality can be locked out. The steel sticks cannot be marked and tracked. Therefore, the available quality information is not related to single sticks, but charges of sticks. For each charge, we know how many sticks had a certain type of error (quality). We want to learn a prediction function from the sensor measurements of the process and the error type counts of charges. The learned model is used to predict the error type for individual sticks at intermediate production stations.

We generalize this learning problem. It deviates from that of supervised learning, where we learn from individually labeled training examples. It is different from semi-supervised learning [5], where we are given at least some labeled examples. Since we have *some* label information, it is not strictly unsupervised

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 349–364, 2011.

learning. Multiple instance learning [23] is a special case, because the bags of examples are either labeled positive or negative, where we have proportions of labels for each charge.

In this paper, we contribute a clustering approach for the problem which has the following properties:

- It empirically shows good prediction performance.
- Learning has linear running time in the number of observations.
- Its prediction models are fast to apply.
- It can handle the case of multiple classes.
- It can handle additional labeled observations, if they exist.
- It can weight the relevance of features.
- It is independent of a certain clustering method.

To the best of our knowledge, no other method exists yet which shares *all* of these properties.

The paper is structured as follows. Section 2 formally defines the learning task and accompanying error measures. We analyze best and worst case from a Bayesian perspective. Section 3 presents a new method for learning from label proportions, LLP. In Sect. 4, we compare the prediction performance and runtime of LLP with other existing methods. In Sect. 5, we shortly discuss related work and conclude.

2 Learning from Label Proportions

In the following, we will first define the task of learning from label proportions. Then, we introduce accompanying measures for evaluating the performance of learners and discuss the problem of model selection. In the last subsection, we explore best and worse case by analyzing the problem from a Bayesian perspective.

2.1 The Learning Task

The task of learning from label proportions can be defined as follows.

Definition 1 (Learning from label proportions). *Let X be an instance space composed of a set of features $X_1 \times \ldots \times X_m$ and $Y = \{y_1, \ldots, y_l\}$ be a set of categorical class labels. Let $P(X, Y)$ be an unknown joint distribution of observations and their class label. Given is a sample of unlabeled observations $U = \{x_1, \ldots, x_n\} \subset X$, drawn i.i.d. from P, partitioned into h disjunct groups $G_1, \ldots, G_h$. Further given are the proportions $\pi_{ij} \in [0, 1]$ of label y_j in group G_i, for each group and label. Based on this information, we seek a function (model) $g : X \rightarrow Y$ that predicts $y \in Y$ for observations $x \in X$ drawn i.i.d. from P, such that the expected error*

$$Err_P = E[L(Y, g(X))] \tag{1}$$

for a loss function $L(Y, g)$ is minimized. The loss penalizes the deviation between the known and predicted label value for an individual observation x.

Labeled examples (unknown)

$G_1 = \{(x_1, 1), (x_3, 1), (x_7, 0)\}$
$G_2 = \{(x_2, 0), (x_4, 0), (x_5, 1), (x_6, 1)\}$
$G_3 = \{(x_8, 0), (x_9, 0)\}$

Sample U (known)

$G_1 = \{x_1, x_3, x_7\}$ $n = 9$
$G_2 = \{x_2, x_4, x_5, x_6\}$ $h = 3$
$G_3 = \{x_8, x_9\}$ $l = 2$

Label proportions (known)

$Y = \{0, 1\}$

$$\prod = \begin{array}{cc} y_1 & y_2 \\ \left(\begin{matrix} 0.33 & 0.67 \\ 0.50 & 0.50 \\ 1.00 & 0.00 \end{matrix}\right) \end{array} \begin{array}{l} \\ |G_1| = 3 \\ |G_2| = 4 \\ |G_3| = 2 \end{array}$$

$$\eta \qquad 0.56 \quad 0.44$$

Fig. 1. Example for a given label proportion matrix Π

The main difference to a supervised learning scenario is that the labels of individual observations are unknown or hidden, i.e., there is *no* set of labeled training instances $(x_i, y_i) \in X \times Y$.

The label proportions π_{ij} can be conveniently written as a $h \times l$ matrix $\Pi = (\pi_{ij})$, where the values in a row $\Pi_{i,\cdot} = (\pi_{i1}, \ldots, \pi_{il})$ sum up to one. (see Fig. 1). The proportion of label y_j over sample U can be calculated from Π:

$$\eta(\Pi, y_j) = \frac{1}{n} \sum_{i=1}^{h} |G_i| \cdot \pi_{ij} \tag{2}$$

By multiplication of π_{ij} with its respective group size $|G_i|$, one gets the frequency counts μ_{ij} of observations with label $y_j \in Y$ in group G_i.

2.2 Training and Test Error

In a supervised learning scenario, a learner can adjust its current hypothesis based on the average loss on the training set. In contrast, when learning from label proportions, one can only measure how well the given proportions are matched. Applying the learned model $g(X)$ to all $x_i \in U$, the resulting label proportions can be calculated, i.e., in each group one counts the number of observations x_i with $g(x_i) = y_j$ for each label $y_j \in Y$ and divides the counts by the size of their respective group. This leads to a new matrix Γ_g, containing the model-based label proportions:

$$\Gamma_g = (\gamma_{ij}^g), \quad \gamma_{ij}^g = \frac{1}{|G_i|} \sum_{x \in G_i} I(g(x), y_j), \quad I = \begin{cases} 1 : g(x) = y_j \\ 0 : g(x) \neq y_j \end{cases} \tag{3}$$

Similarly to defining a loss function for individual observations, it is now possible to define a loss function for individual matrix entries, for example by the squared error $(\pi_{ij} - \gamma_{ij}^g)^2$. The total deviance between Π and Γ_g can then be defined as the average squared error over all matrix entries:

$$\mathrm{Err}_{MSE}(\Pi, \Gamma_g) = \frac{1}{hl} \sum_{i=1}^{h} \sum_{j=1}^{l} (\pi_{ij} - \gamma_{ij}^g)^2 \tag{4}$$

Calculating Err_{MSE} for a function $g(X)$ on sample U might be seen as an analogon to the training error in supervised learning. However, the loss in Err_{MSE} uses aggregated label information, although by Definition 1, we really need to minimize the loss over individual observations. The mismatch between the two measures can lead to problems, because many labelings of U can minimize Err_{MSE}. For example, when randomly sampling μ_{ij} many observations from G_i and assigning them label y_j, the model-based label proportions will always match exactly the given proportions. Hence, labelings that minimize Err_{MSE} don't necessarily minimize the average loss over individual observations, already on sample U. Therefore, obtaining a good estimate of Err_P is difficult. In supervised learning, one may select the model which has the lowest average loss over one or several test sets. But without labels for individual observations in the test set, only knowing its label proportions, a low Err_{MSE} on the test set is no reliable indicator for a good model. As for the training error, many different labelings can lead to the same label proportions, but only a few labelings will minimize Err_P. However, if given a labeled test set, it is possible to evaluate different models as in the supervised case.

For the experiments in Sect. 4, the error between Π and Γ_g wasn't measured by Err_{MSE}, but by Err_{Π}, a combination of two different error measures:

$$\mathrm{Err}_{\Pi}(\Gamma_g) = \sqrt{\mathrm{Err}_{weight}(\Pi, \Gamma_g) \cdot \mathrm{Err}_{prior}(\Pi, \Gamma_g)} \quad \text{with} \tag{5}$$

$$\mathrm{Err}_{weight}(\Pi, \Gamma_g) = \frac{1}{hl} \sum_{i=1}^{h} \sum_{j=1}^{l} \eta(\Pi, y_j) \frac{|G_i|}{n} (\pi_{ij} - \gamma_{ij}^g)^2 \quad \text{and} \tag{6}$$

$$\mathrm{Err}_{prior}(\Pi, \Gamma_g) = \frac{1}{l} \sum_{j=1}^{l} \left(\eta(\Pi, y_j) - \eta(\Gamma_g, y_j)\right)^2 \tag{7}$$

Err_{weight} weights the squared error of individual matrix entries by their relative group and class size. Err_{prior} catches situations where two hypotheses g_1 and g_2 are indistinguishable from each other, because the total error sum over all matrix entries is the same. In such cases, they may be distinguished by their column differences, as calculated by η.

If in addition to the label proportions, the labels $c_1, \ldots, c_t$ of individual observations $T = \{a_1, \ldots, a_t\} \subseteq U$ are given, error criterion (5) can be easily extended by including the average loss Err_T over these training examples:

$$\mathrm{Err}_{\Pi} = \sqrt{\mathrm{Err}_{weight} \cdot \mathrm{Err}_{prior} \cdot \mathrm{Err}_T} \quad \text{with} \quad \mathrm{Err}_T = \frac{1}{t} \sum_{i=1}^{t} L(c_i, g(a_i)) \tag{8}$$

2.3 Best and Worst Case from a Bayesian Perspective

From a Bayesian perspective, a good model can be obtained by estimating the conditional class density $P(Y|X)$. Applying Bayes theorem, one recognizes that $P(Y|X)$ may also be estimated from other unknown densities—the class-conditional density $P(X|Y)$ and the class prior density $P(Y)$:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{9}$$

$P(X)$ doesn't need to be estimated, since it can be calculated from $P(X|Y)$ and $P(Y)$. When learning from label proportions, the class prior $P(y_j)$ for label y_j can be estimated as $\eta(\Pi, y_j)$, the proportion of y_j over sample U. However, finding a good estimate for $P(X|Y)$ depends on the distribution of observations over the given groups and the form of matrix Π.

In the *best case*, each group G_i only contains observations from a single class and at least l groups contain observations from different classes. If $\pi_{ij} = 1$ appears in a row, all observations in the group must have the same label, which can be assigned to all group members. We then are in a familiar supervised learning scenario and can choose from many well-known classifiers for training.

However, without further knowledge about the distribution of observations over the groups, the best we can assume is a uniform distribution. Here, in the *worst case*, all π_{ij} are equal. Then, if we interpret π_{ij} as an estimate for the class prior $P(y_j|G_i)$ of group G_i, it equals the estimated class prior $P(y_j)$. Since each observation has the same probability of being sampled into group G_i, and we assumed all priors to be equal, we can only guess the correct label with probability $1/l$. In general, if the number of groups remains constant, $P(y_j|G_i)$ will approximate $P(y_j)$ for large sample sizes n. This can be seen if we imagine each group G_i to be an independent data set with observations sampled from the same distribution $P(X, Y)$.

For cases where all $P(y_j)$ are different, a better performance can be achieved than in the worst case. One can at least predict the majority class. The question is if one can get any better. Except for the best case, the estimation of $P(X|Y)$ is difficult, because observations with the same label are spread over all groups. The LLP algorithm, introduced in the next section, is based on the idea that observations sharing the same label might also have similar feature values.

3 Learning from Label Proportions by Clustering

The k-Nearest-Neighbour classifier predicts the majority label of k known observations closest to a given search point. It is presupposed that observations lying close together in local regions of the input space also share the same class label. If we could somehow identify these local groups of observations, which is the problem of cluster analysis, the only problem left was to assign the correct labels to the clusters.

Definition 2 (Cluster analysis). *Given a sample U of n unlabeled observations $x_1, \ldots, x_n$ and a measure $d : X \times X \rightarrow \mathbb{R}^+$ for the dissimilarity of observations, the aim of cluster analysis is to determine a set $\mathcal{C} = \{C_1, \ldots, C_k\}$ (clustering) of subsets $C_i \subset U$ (clusters), such that observations within the same* *cluster are more similar to each other than those in different clusters, as measured by a quality function* $q : 2^{2^X} \to \mathbb{R}^+$.

Many algorithms have been developed for solving this task. We focus on those returning disjunct clusters, like the well-known k-Means algorithm [16], which was also used for the experiments in Sect. 4. Given a clustering, it must be found out which cluster best represents which class. The problem is solved by assigning each cluster a label such that Err_{Π} (5) is minimized (see Sect. 3.3).

In how far similar observations share the same class label not only depends on $P(X, Y)$, but also on the chosen similarity measure. According to Hastie et al. [13], the relevance of features can have an enormous influence on the clustering results. Therefore, the similarity measure should respect weights $w_f \in [0, 1]$ for each feature, as given by a vector $\boldsymbol{w} = (w_1, \ldots, w_m)$. In unsupervised learning, such weights are usually specified by a domain expert. Here, the relevance weights can be approximated automatically (see Sect. 3.2), based on criterion Err_{Π}. In the next section, the accompanying optimization problem is stated. Then, the LLP algorithm for solving it is described.

3.1 Optimization Problem

Let the vector $\boldsymbol{\lambda}_{\mathcal{C}} = (\lambda_1, \ldots, \lambda_k)$ with $\lambda_j \in Y$ represent a labeling for a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$. Let $m_{\boldsymbol{\lambda}_{\mathcal{C}}} : U \to Y$ be a mapping that returns for a given observation $x \in C_i$ the label λ_i. Given a clustering $\mathcal{C}$, we are searching for a labeling $\boldsymbol{\lambda}_{\mathcal{C}}$ of the clusters that minimizes the error (5) between the model-based label proportions $\Gamma_{m_{\boldsymbol{\lambda}_{\mathcal{C}}}}$ and the known label proportions:

$$\min_{\boldsymbol{\lambda}_{\mathcal{C}}} \mathrm{Err}_{\Pi}(\Gamma_{m_{\boldsymbol{\lambda}_{\mathcal{C}}}}) \tag{10}$$

Let $q_{\boldsymbol{w}}$ be a function which is able to assess the quality of a clustering based on a similarity measure that respects feature weights. We are searching for a clustering which maximizes $q_{\boldsymbol{w}}$ and whose labeling most minimizes Err_{Π}, for all possible weight vectors $\boldsymbol{w}$. This optimization problem can be stated as follows:

$$\min_{\boldsymbol{w}} \mathrm{Err}_{\Pi}(\Gamma_{m_{\boldsymbol{\lambda}^*_{\mathcal{C}}}}), \quad \boldsymbol{\lambda}^*_{\mathcal{C}} = \operatorname*{argmin}_{\boldsymbol{\lambda}_{\mathcal{C}^*}} \mathrm{Err}_{\Pi}(\Gamma_{m_{\boldsymbol{\lambda}_{\mathcal{C}^*}}}), \quad \mathcal{C}^* = \operatorname*{argmax}_{\mathcal{C}} q_{\boldsymbol{w}}(\mathcal{C}) \tag{11}$$

3.2 The LLP Algorithm

The LLP algorithm solves problem (11) by an evolutionary strategy. For each weight vector $\boldsymbol{w}$, the sub-optimization problem of maximizing $q_{\boldsymbol{w}}$ is solved by an inner clustering algorithm, where the particular $q_{\boldsymbol{w}}$ depends on the algorithm. The only prerequisite for the clusterer is that it returns disjunct clusters and respects different feature weights. The sub-optimization problem (10) is independent from the clusterer and currently can be solved by two different labeling heuristics introduced in Sect. 3.3. Using an evolutionary strategy as a wrapper has the advantage that it is not necessary to integrate criterion Err_{Π} into the

Algorithm 1. The LLP algorithm

```
Input: Label proportion matrix $\Pi$, sample $U$, groups $\mathcal{G} = \{G_1, \ldots, G_h\}$,
  labels $Y = \{y_1, \ldots, y_l\}$, clustering algorithm clusterer, labeling algorithm labeler,
  parameters maxgen, psize, mutvar, crossprob, tournsize
Output: Clustering $\mathcal{C}$, labeling $\boldsymbol{\lambda}_\mathcal{C}$, weight vector $\boldsymbol{w}$
  $best\_fit := -\infty$; $generation := 0$
  Randomly initialize a population $P$ of psize normalized weight vectors
  while $generation < maxgen$ do
    for $\boldsymbol{w} \in P$ do
      $\mathcal{C} := clusterer(\ U,\ \boldsymbol{w}\ )$
      $(\boldsymbol{\lambda}_\mathcal{C},\ \mathrm{Err}_\Pi) := labeler(\ \mathcal{C},\ \mathcal{G},\ \Pi,\ Y\ )$
      if $best\_fit < -\mathrm{Err}_\Pi$ then
        $best\_fit := -\mathrm{Err}_\Pi$;  $best\_\mathcal{C} := \mathcal{C}$;  $best\_\boldsymbol{\lambda}_\mathcal{C} := \boldsymbol{\lambda}_\mathcal{C}$;  $best\_\boldsymbol{w} := \boldsymbol{w}$
      end if
    end for
    $generation := generation + 1$
    if $generation < maxgen$ then
      $Pcopy := P$
      Gaussian mutation of all individual weights in $Pcopy$ with variance $mutvar$
      $Pchildren :=$ Uniform crossover on $P \cup Pcopy$ with probability $crossprob$
      $P :=$ Tournament selection with size $tournsize$ on $P \cup Pcopy \cup Pchildren$
    end if
  end while
  return $best\_\mathcal{C}$, $best\_\boldsymbol{\lambda}_\mathcal{C}$, $best\_\boldsymbol{w}$
```

optimization problem of the inner clustering algorithm. For example, we already have run LLP successfully with Kernel k-Means [10], DBSCAN [12] and PROCLUS [1], without modification. Moreover, LLP can be used with different error measures, for instance with criterion (8) that can respect individually labeled examples.

LLP (see Alg. 1) takes a clustering algorithm *clusterer* and a labeling algorithm *labeler* as parameters, in addition to Π, U, $G_1, \ldots, G_h$ and Y, which are related to the label proportions learning task. LLP then approximates the optimal weight vector $\boldsymbol{w}$ and returns $\boldsymbol{w}$, the related clustering $\mathcal{C}$ and labels $\boldsymbol{\lambda}_\mathcal{C}$ for the clusters.

The evolutionary strategy starts with a random population P of normalized weight vectors, $\boldsymbol{w}_i \in [0, 1]$. For each individual in P, the clustering algorithm *clusterer* is called. The clusters are labeled according to the given labeling algorithm *labeler* and the fitness is evaluated by criterion Err_Π. If the fitness is higher than the best fitness seen so far, the newly found clustering, labeling and weight vector are memorized as the new best ones. In each generation, the weight values in a copy of P are mutated by a Gaussian distribution and, with a certain probability, exchanged with P by a crossover operator. Then, the individuals take part in a tournament and only the best ones are kept in the next generation. This process is repeated until the maximum number of generations as specified by the user is reached.

Algorithm 2. The greedy labeling algorithm

```
Input: Clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$, groups $\mathcal{G} = \{G_1, \ldots, G_h\}$,
  label proportion matrix $\Pi$, labels $Y = \{y_1, \ldots, y_l\}$
Output: Labeling $\boldsymbol{\lambda}_{\mathcal{C}} = (\lambda_1, \ldots, \lambda_k)$
  Initialize components of $\boldsymbol{\lambda}_{\mathcal{C}}$ with $y_1$;
  for $i := 1$ to $k$ do
    lowest_error := 0; best_label := $y_1$;
    for $j := 1$ to $l$ do
      $\boldsymbol{\lambda}_{\mathcal{C}}[i] := y_j$;
      $\Gamma_m$ := count_labels( $\mathcal{G}$, $\mathcal{C}$, $\boldsymbol{\lambda}_{\mathcal{C}}$ );
      current_error := $\mathrm{Err}_{\Pi}(\Gamma_m)$;
      if current_error < lowest_error then
        lowest_error := current_error;
        best_label := $y_j$;
      end if
    end for
    $\boldsymbol{\lambda}_{\mathcal{C}}[i]$ := best_label;
  end for
  return $\boldsymbol{\lambda}_{\mathcal{C}}$
```

3.3 Labeling Heuristics

The following two labeling algorithms solve the sub-optimization problem (10) heuristically.

Greedy Labeling. As shown in Alg.2, in the initial step, all clusters get label y_1. Then, consecutively for each cluster, we calculate Γ_m for all labels and memorize the label that most reduces $\mathrm{Err}_{\Pi}(\Gamma_m)$. The strategy has runtime $k \cdot l$.

Exhaustive Labeling. Since k can be restricted to a small number and $l = 2$ for a binary classification problem, trying l^k possible labelings for a clustering $\mathcal{C}$ is no problem. In our experiments (see section 4), good solutions often were found for $k \leq 6$. For each labeling, we need to calculate Err_{Π}, which takes linear time in the number of observations n. The calculations only involve basic operations like count, addition, multiplication and division (see (5)).

3.4 Run-Time Analysis

The user-specified parameters *maxgen*, *psize* and *tournsize* are constants. They do not depend on the number of observations n and limit the number of iterations to be constant. As discussed in Sect. 3.3, the asymptotic run-time of the heuristic labeling strategies is linear in n, as k and l are constants and the evaluation of (5) takes linear time. The asymptotic run-time of LLP will otherwise depend on the used cluster algorithm. For example, if we allow for approximate solutions and limit the number of iteration steps, k-Means has linear run-time. Hence, LLP has linear run-time.

Table 1. UCI data sets used for the experiments

Dataset	n	m	Dataset	n	m
CREDITA	690	42	SONAR	208	60
VOTE	435	16	DIABETES	768	8
COLIC	368	60	BREAST CANCER	286	38
IONOSPHERE	351	34	HEARTC	303	22

3.5 Generating a Prediction Model

The LLP algorithm returns labeled clusters of sample U. It is then possible to assign labels to individual observations $x_i \in U$ with $m_{\boldsymbol{\lambda}_C}$. To predict the labels of new observations, the clustering must be transformed into a prediction model. The way to do this depends on the used clustering algorithm. In the case of k-Means, one can simply assign new observations to their closest cluster mean and predict the corresponding cluster label. A big advantage of the cluster mean model is that it is usually very small, as $k \ll n$, and therefore fast to apply. Another option for getting a prediction model is to train a classifier like Naïve Bayes [14] or a Support Vector Machine [22] in a subsequent step, based on the now labeled observations. However, this increases the training time.

4 Experiments

We have compared the LLP algorithm to three state-of-the-art methods for learning from label proportions: The Mean Map method [19], Inverse Calibration (Invcal) [21] and AOC Kernel k-Means (AOC-KK) [6]. For a further discussion of these methods, see Sect. 5. LLP has been implemented in Java. As inner clustering algorithm, we have used Fast k-Means [11], which is a variant of k-Means utilizing the triangle inequality for faster distance calculations. As distance measure, we have used the weighted Euclidean distance. We have implemented AOC-KK using a combination of Java and Matlab. For Mean Map and Invcal, we used R scripts provided by the author of Invcal [21].

4.1 Prediction Performance Experiments

The prediction performance (accuracy) of LLP, AOC-KK, Invcal and Mean Map has been assessed on the eight UCI [3] data sets shown in Table 1. We have mapped each possible value of a nominal feature to a binary numerical feature with values 0 or 1. Numerical features were normalized to the $[0, 1]$ interval. Table 1 shows the number of features m after this preprocessing step.

In each single experiment, the accuracy has been assessed by a 10-fold cross-validation. For learning from label proportions, we have partitioned the training set of a particular fold into groups of size σ, by uniform sampling of observations. We tried several group sizes σ: 2, 4, 8, 16, 32, 64 and 128 (with the last group smaller than σ, if necessary). The label proportions were calculated and the individual labels removed. In each fold, the accuracy of the learned prediction model has been calculated on a labeled test set.

The kernel methods Mean Map, Invcal and AOC-KK have been tested with the linear kernel, polynomial kernels of degree 2 and 3 and radial basis kernels ($\gamma = 0.01$, 0.1 and 1.0). LLP has been tested with both labeling heuristics (see Sect. 3.3), for cluster sizes $k \in [2, 12]$. As parameters for the evolutionary strategy, we used $maxgen = 10$, $psize = 25$, $mutvar = 1.0$, $crossprob = 0.3$ and $tournsize = 0.25$. By running LLP with k-Means, we get a prediction model consisting of cluster means. The same is true for AOC-KK. However, the cluster methods also assign labels to each observation in sample U, allowing for a subsequent training of other classifiers. Based on the clustering results, we have trained models for Naïve Bayes [14], kNN [2], Decision Trees [20], Random Forests [4], and the SVM [22] with linear and radial basis kernel. The model parameters have been optimized by a grid or evolutionary search.

The combination of all datasets, group sizes, classifiers, their variants and parameters results in a total of 13.216 experiments: 672 for Mean Map and Invcal, 2.688 for AOC-KK and 9.856 for LLP. For group sizes 16, 32, 64 and 128 on the datasets COLIC and SONAR, and for group size 128 on CREDITA, we conducted additional experiments with LLP for $maxgen = 5$ and $psize = 100$. In some cases, we got a better prediction accuracy. All experiments took about three weeks. They were run in parallel on up to six machines with an AMD Dual-Core or Quad-Core Opteron 2220 processor and a maximum of 4 GB main memory.

4.2 Prediction Performance Results

Figure 2 contains plots of the highest achieved accuracies for all data sets and group sizes, based on the best performing models of LLP, AOC-KK, Invcal and MeanMap, over all conducted experiments. LLP shows a higher accuracy than Invcal for many group sizes on the data sets CREDITA, VOTE, COLIC, SONAR and BREAST CANCER. On CREDITA, VOTE, IONOSPHERE, SONAR and DIABETES, the variance of accuracy between group sizes is smaller for LLP in comparison to the other methods. Mean Map performs worse than LLP and Invcal in many cases. The performance of AOC-KK varies, depending on the data set. It shows good performance on BREAST CANCER and HEARTC, but not on the others. Except for the BREAST CANCER and VOTE data set and a few other accuracy values, the overall accuracy of *all* methods decreases with a larger group size.

4.3 Statistical Significance

For the comparison of multiple classifiers over multiple data sets, Demsar [9] proposes the Friedman test, which is a non-parametric equivalent of ANOVA. We use the adjusted version, with a test statistic distributed according to the F-distribution (see [9]). The test ranks the classifiers for each data set separately. Under the null-hypothesis, the average ranks of the classifiers should be equal. In

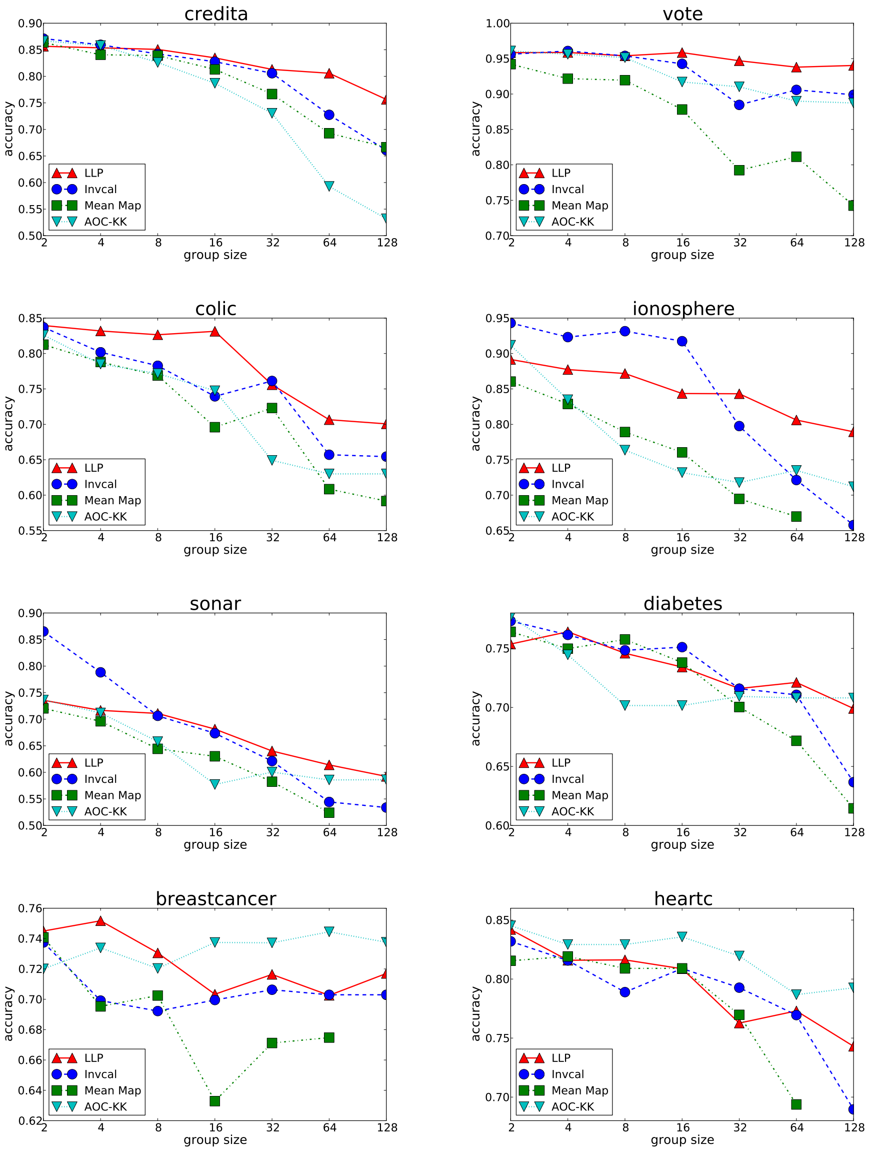


Fig. 2. Highest accuracies for all data sets and group sizes, over all 13.216 runs of LLP, AOC-KK, Invcal and MeanMap (plus the additional runs of LLP with $maxgen = 5$ and $psize = 100$). Some values for Mean Map and group size 128 are missing in the plots, due to an error in the R script.

Table 2. Average ranks of classifiers by group size, and their difference to LLP's rank, based on the best models for each data set and group size. Positive difference values indicate a better performance of LLP. Highest ranks and significant differences (higher than CD) at the 10%-level are marked in bold.

σ	2	4	8	16	32	64	128
	Average Ranks						
LLP	2.500	**1.875**	**1.500**	**1.875**	**1.625**	**1.375**	**1.375**
AOC-KK	**2.000**	2.750	3.000	2.875	2.625	2.375	2.000
Invcal	**2.000**	**1.875**	2.375	2.125	2.125	2.275	2.625
Mean Map	3.500	3.500	3.125	3.125	3.625	3.875	-
	Differences, $CD_{<128}$=1.4317, CD_{128}=0.98						
AOC-KK	-0.500	0.875	**1.500**	1.000	1.000	1.000	0.625
Invcal	-0.500	0.000	0.875	0.250	0.500	1.000	**1.250**
Mean Map	1.000	**1.625**	**1.625**	1.250	**2.000**	**2.500**	-

case of a critical difference, the null-hypothesis can be rejected. The test yielded significant differences for all group sizes. One can then proceed with a post-hoc test. We have decided for the two-tailed Bonferroni-Dunn test (again, see [9]), which is for comparing a single classifier (here, LLP) to all others.

Table 2 shows the average ranks of the compared classifiers and their difference to LLP's rank. Each rank was calculated based on the best performing models (including the standard classifiers), over all conducted experiments. The table also shows the critical difference (CD) values for the Bonferroni-Dunn test. The CD for $\sigma = 128$ is different, because Mean Map was not included in the comparison, due to missing values. LLP has the highest rank in six cases, for $\sigma > 2$. At the 10%-level, LLP is significantly better than AOC-KK for $\sigma = 8$, better than Invcal for $\sigma = 128$ and better than Mean Map for $\sigma = 4, 8, 32$ and 64. In all other cases, LLP performs equivalently.

The ranks in Table 3 are based on different models than those in Table 2. For LLP and AOC-KK, we have only included the best performing cluster mean models. We have compared them to the best performing models of Invcal and Mean Map, i.e. to different kernels. The cluster mean models perform significantly better than Mean Map for $\sigma = 64$ and better than Invcal for $\sigma = 128$. In all other cases, they show an equivalent prediction performance, but are faster to train and apply, as discussed in Sects. 3.4 and 4.4. In the same way, we have separately compared the exhaustive and greedy labeling strategies to the best performing models of all other classifiers. The exhaustive strategy performed better, in the sense that it showed more significant differences to the other methods.

Concerning the performance and significance of the standard classifiers, which were trained based on the LLP and AOC-KK cluster models, Decision Trees performed significantly better than Invcal for $\sigma = 128$, better than Mean Map for $\sigma = 64$ and better than AOC-KK for $\sigma = 4$ and 32. Naive Bayes, k-NN and Random Forests had a performance similar to the cluster mean models.

Table 3. Average ranks of classifiers by group size, and their difference to LLP's rank. Ranks are based on the best performing models of Invcal and Mean Map and the best performing cluster mean models of LLP and AOC-KK. Positive difference values indicate a better performance of LLP. Highest ranks and significant differences (higher than CD) at the 10%-level are marked in bold.

σ	2	4	8	16	32	64	128
	Average Ranks						
LLP	2.375	2.375	**2.000**	2.250	2.250	**1.750**	**1.375**
AOC-KK	3.750	3.250	3.125	2.875	2.875	2.375	2.125
Invcal	**2.125**	**1.625**	2.125	**1.750**	**1.625**	2.000	2.500
Mean Map	2.750	2.750	2.750	3.125	3.250	3.875	-
	Differences, $CD_{<128}$=1.4317, CD_{128}=0.98						
AOC-KK	1.375	0.875	1.125	0.625	0.625	0.625	0.750
Invcal	-1.000	-0.750	0.125	-0.500	-0.625	0.250	**1.125**
Mean Map	0.125	0.375	0.750	0.875	1.000	**2.125**	-

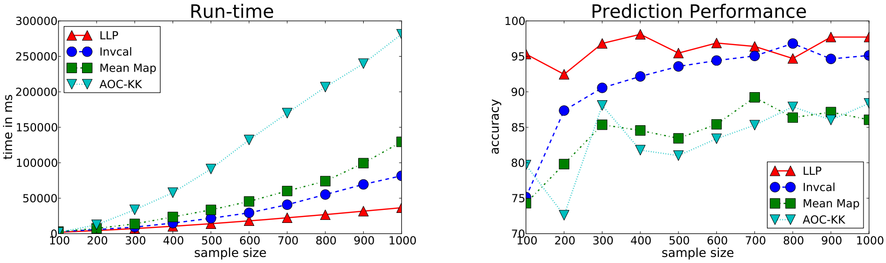


Fig. 3. Average run-time and accuracy of 10-fold cross-validations with LLP, Invcal, Mean Map and AOC-KK on several samples of random data. The data was generated for a two Gaussian mixture classification problem (n = 10000, m = 10, feature values normalized to $[0, 1]$).

The linear SVM and the SVM with radial basis kernels showed no significant differences to Invcal, MeanMap or AOC-KK.

4.4 Run-Time Comparison

For an empirical run-time comparison of LLP, Invcal, Mean Map and AOC-KK, we have generated random data for a two Gaussian mixture classification problem (10.000 observations and 10 features, with values normalized to $[0, 1]$). Then, the average run-time for training and the accuracy of the classifiers over 10 folds of a cross-validation has been assessed for different samples of the data, with varying sizes (see Fig. 3). The group size for learning from label proportions has been $\sigma = 16$ for all runs. A radial basis kernel with $\gamma = 0.1$ has been used for the kernel methods. LLP has been run with the exhaustive labeling strategy and Fast k-Means ($k = 6$), with parameters $maxgen = 3$, $psize = 25$, $mutvar = 1.0$, *crossprob* = 0.3 and *tournsize* = 0.25 for the evolutionary optimization. Both LLP and AOC-KK used the cluster mean model for prediction.

LLP shows a high prediction performance for all sample sizes. Moreover, LLP has the lowest run-time. However, since the methods are implemented in different programming languages (Java, Matlab, R), one should not compare the absolute times, but the slope of the curves. The curve of LLP's run-time is a straight line, while the other curves indicate a polynomial run-time.

5 Related Work

The problem of learning from label proportions has gained attention in the machine learning community, only recently. Musicant et al. [18] formally defined the problem of learning from aggregate values for regression and classification tasks. They modified well-known methods like k-NN [2], backpropagation neural networks [17] and the linear SVM [22] to respect the given label proportions. Their experimental results focus on regression tasks, while we are mainly interested in classification.

Quadrianto et al. [19] have proposed the Mean Map method which estimates the conditional class probability $P(Y|X,\theta)$ by conditional exponential models, using a feature map $\Phi(X,Y)$ and a normalization function g:

$$P(Y|X,\theta) = \exp(\, \langle \Phi(X,Y), \theta \rangle - g(\theta|X)\,) \tag{12}$$

The parameter θ is estimated by solving a convex maximization problem for the conditional log-likelihood $\log P(Y|X,\theta)$. This depends on the unknown labels only in terms of the empirical mean μ_{XY}, which they approximate by the observation means for each group and its given label proportions. They compare Mean Map to kernel density estimation, discriminative sorting, and MCMC [15]. Mean Map outperformed the related techniques. For this reason, we have compared LLP only to Mean Map. Although LLP and Mean Map can both handle multi class problems, for easier comparison with Inverse Calibration, we have restricted our experiments to binary classification problems. During training, Mean Map needs to solve a general convex optimization problem. In contrast, LLP's worst-case training time is linear in n for the cluster mean models. As was shown in Sect. 4, these models achieved equivalent accuracy. Moreover, over all trained models, LLP's accuracy has been significantly higher than Mean Map's for several group sizes.

Rueping [21] proposes the inverse calibration method. The author converts the regression SVM (SVR) into a probabilistic classifier by applying a scaling function σ to the outputs $y = f(x)$, such that $\sigma(y)$ is a good estimate for $p = P(y = 1|x)$. Since no individual estimates p for each observation x are given, it is only required that f predicts $y = \sigma^{-1}(p)$ well on average. This is equivalent to approximating the given label proportions well. The constraints are integrated as auxiliary conditions into the standard SVR optimization problem. LLP outperformed the inverse calibration for $\sigma = 128$, also with the cluster mean models. It achieved equivalent results on smaller group sizes, but in shorter time.

For a semi-supervised learning case, Dara et al. [7] cluster the data first with SOMs and then label the clusters. However, they have labeled observations, which we do not. Demiriz et al. [8] adapt the k-Means optimization problem to respect labeled data. Again, this is a semi-supervised setting, with labeled observations. The idea is similar though to the AOC Kernel k-Means algorithm by Chen et al. [6], who integrate the loss function (4) into the optimization problem of Kernel k-Means clustering [10]. In comparison to AOC-KK, LLP has achieved significantly better accuracy for $\sigma = 8$ over all conducted experiments. For $\sigma > 2$, LLP had a higher average rank than AOC-KK. LLP needs only linear training time, while in contrast, AOC-KK solves a quadratic optimization problem in each iteration step of Kernel k-Means.

6 Conclusions and Future Work

We have presented a new approach for learning from label proportions, the LLP algorithm. With k-Means as the clustering algorithm, LLP has only linear worst-case training time and its cluster mean models are small and fast to apply. In comparison to state-of-the-art methods, which need more training time, the cluster mean models have shown significantly better or equivalent prediction accuracy. By training other classifiers on the labeled clusters, the highest achieved accuracy of LLP was significantly different in even more cases, and LLP had the highest average rank for all $\sigma > 2$. In the future, we want to evaluate LLP's performance on data from the steel factory, as mentioned in the introduction. Moreover, it would be interesting to assess LLP's prediction performance with multi class problems and additional labeled observations. Another direction is to use different clustering algorithms with LLP and compare their performance.

Acknowledgements. This work has been supported by the DFG, Collaborative Research Center SFB876, project B3.

References

1. Aggarwal, C.C., Wolf, J.L., Yu, P.S., Procopiuc, C., Park, J.S.: Fast algorithms for projected clustering. In: Proc. of the Int. Conf. on Management of Data, SIGMOD 1999, pp. 61–72. ACM, New York (1999)
2. Aha, D.: Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms. Int. J. of Man-Machine Studies 36(2), 267–287 (1992)
3. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
4. Breimann, L.: Random forests. Machine Learning 45, 5–32 (2001)
5. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
6. Chen, S., Liu, B., Qian, M., Zhang, C.: Kernel k-Means based framework for aggregate outputs classification. In: Proc. of the Int. Conf. on Data Mining Workshops (ICDMW), pp. 356–361 (2009)
7. Dara, R., Kremer, S., Stacey, D.: Clustering unlabeled data with SOMs improves classification of labeled real-world data. In: Proc. of the 2002 Int. Joint Conf. on Neural Networks (IJCNN), vol. 3, pp. 2237–2242 (2002)

8. Demiriz, A., Bennett, K., Bennett, K.P., Embrechts, M.J.: Semi-supervised clustering using genetic algorithms. In: Proc. of Artif. Neural Netw. in Eng (ANNIE), pp. 809–814. ASME Press (1999)
9. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
10. Dhillon, I., Guan, Y., Kulis, B.: Kernel k-Means: spectral clustering and normalized cuts. In: Proc. of the 10th Int. Conf. on Knowl. Discov. and Data Mining, SIGKDD 2004, pp. 551–556. ACM, New York (2004)
11. Elkan, C.: Using the triangle inequality to accelerate k-means. In: Proc. of the 20th Int. Conf. on Machine Learning (ICML) (2003)
12. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the 2nd Int. Conf. on Knowl. Discov. and Data Mining, pp. 226–231. AAAI Press, Menlo Park (1996)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Statistics, 2nd edn. Springer, Heidelberg (2009)
14. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proc. of the 11th Conf. on Uncertainty in Artif. Int., pp. 338–345. Morgan Kaufmann, San Francisco (1995)
15. Kueck, H., de Freitas, N.: Learning about individuals from group statistics. In: Uncertainty in Artif. Int. (UAI), pp. 332–339. AUAI Press, Arlington (2005)
16. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Symp. Math. Stat. & Prob., pp. 281–297 (1967)
17. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
18. Musicant, D.R., Christensen, J.M., Olson, J.F.: Supervised learning by training on aggregate outputs. In: Proc. of the 7th Int. Conf. on Data Mining (ICDM), pp. 252–261. IEEE Computer Society, Washington, DC, USA (2007)
19. Quadrianto, N., Smola, A.J., Caetano, T.S., Le, Q.V.: Estimating labels from label proportions. J. Mach. Learn. Res. 10, 2349–2374 (2009)
20. Quinlan, J.R.: Induction of decision trees. Machine Learning 1(1), 81–106 (1986)
21. Rüping, S.: SVM classifier estimation from group probabilities. In: Proc. of the 27th Int. Conf. on Machine Learning (ICML) (2010)
22. Vapnik, V.: The nature of statistical learning theory, 2nd edn. Springer, New York (1999)
23. Witten, I.H., Eibe, F., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques. In: Data Management Systems, 3rd edn. Elsevier, Inc., Burlington (2011)

The Minimum Code Length for Clustering Using the Gray Code

Mahito Sugiyama[1,2] and Akihiro Yamamoto[1]

[1] Graduate School of Informatics, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
mahito@iip.ist.i.kyoto-u.ac.jp,
akihiro@i.kyoto-u.ac.jp
[2] Research Fellow of the Japan Society for the Promotion of Science

Abstract. We propose new approaches to exploit compression algorithms for clustering numerical data. Our first contribution is to design a measure that can score the quality of a given clustering result under the light of a *fixed* encoding scheme. We call this measure the *Minimum Code Length* (MCL). Our second contribution is to propose a general strategy to translate any encoding method into a cluster algorithm, which we call COOL (COding-Oriented cLustering). COOL has a low computational cost since it scales linearly with the data set size. The clustering results of COOL is also shown to minimize MCL. To illustrate further this approach, we consider the *Gray Code* as the encoding scheme to present G-COOL. G-COOL can find clusters of arbitrary shapes and remove noise. Moreover, it is robust to change in the input parameters; it requires only two lower bounds for the number of clusters and the size of each cluster, whereas most algorithms for finding arbitrarily shaped clusters work well only if all parameters are tuned appropriately. G-COOL is theoretically shown to achieve internal cohesion and external isolation and is experimentally shown to work well for both synthetic and real data sets.

Keywords: Clustering, Compression, Discretization, Gray code.

1 Introduction

Clustering is a fundamental task in data analysis, and many clustering algorithms have been developed in the fields of machine learning and knowledge discovery [1,9,14]. Several clustering algorithms have been recently proposed that focus on the *compression* of data points.

Kontkanen *et al.* [19] proposed the *minimum description length* (MDL) approach to clustering taking advantage of an information theoretic framework. However, data encoding has to be optimized to find the best clusters; that is, all encoding schemes are considered within the clustering process under the MDL criterion. As a result, their approach takes quadratic time with respect to the data set size and can only be handled in practice using a stochastic algorithm [18]. Cilibrasi and Vitányi [6] proposed a clustering algorithm based on the *Kolmogorov complexity*. Since their method measures the distance between two data points on the basis of compression of finite sequences (*i.e.*,

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 365–380, 2011.

discrete variables), it is difficult to apply it to multivariate data of continuous variables. Moreover, although there are other approaches [16,20] that focus on compression of data, they perform simple agglomerative hierarchical clustering, so it takes quadratic time with respect to the data set size. These approaches are therefore not suitable for clustering massive data sets.

Here we propose a new measure, called the *minimum code length* (MCL), to score the quality of a given clustering result under a *fixed* encoding scheme. This use of fixed encoding enables the performance of fast (*i.e.*, linear complexity with respect to the data set size) and exact clustering since we do not need to optimize data encoding. We present a clustering algorithm, called COOL (CODing-Oriented cLustering), that always finds the best clusters; *i.e.*, the globally optimal clusters which minimizes MCL, and requires only the lower bounds for the number and size of the clusters. The discretization of continuous variables with the fixed encoding scheme coincides with the clustering process itself — a hierarchy of clusters is introduced automatically by increasing the accuracy of discretization.

Mathematically, an encoding, or *embedding*, is a mapping from real numbers to infinite sequences over some alphabet [29], and discretization is realized by truncation of infinite sequences. For example, in the binary embedding γ_{B}, every real number in $[0, 1]$ is translated into an infinite sequence composed of $\mathtt{0}$ and $\mathtt{1}$; *e.g.*, $\gamma_{\mathrm{B}}(0) = \mathtt{000}\ldots$, $\gamma_{\mathrm{B}}(0.2) = \mathtt{001}\ldots$, and $\gamma_{\mathrm{B}}(0.4) = \mathtt{011}\ldots$, where the first bit is $\mathtt{0}$ if the value is in the interval $[0, 0.5]$ and $\mathtt{1}$ if in $(0.5, 1]$. If these sequences are truncated at the first bit, all of them become $\mathtt{0}$, and hence they are considered as in the same cluster since they cannot be distinguished. If they are then truncated at the second bit, both 0 and 0.2 become $\mathtt{00}$, and 0.4 becomes $\mathtt{01}$. Thus, two clusters are generated: $C_1 = \{0, 0.2\}$ and $C_2 = \{0.4\}$. This means that representatives of C_1 and C_2 are $\mathtt{00}$ and $\mathtt{01}$, respectively. Finally, if they are truncated at the third bit, 0 and 0.2 are separated. The hierarchy is therefore constructed as $\{\{0, 0.2, 0.4\}\}$, $\{\{0, 0.2\}, \{0.4\}\}$, and $\{\{0\}, \{0.2\}, \{0.4\}\}$.

The complexity of making clusters can be measured by the length of the cluster representatives. In the above example, $2 + 2 = 4$ for clusters $\{0, 0.2\}$ and $\{0.4\}$ ($\mathtt{00}$ and $\mathtt{01}$), and $3 + 3 + 2 = 8$ for $\{0\}$, $\{0.2\}$, and $\{0.4\}$ ($\mathtt{000}$, $\mathtt{001}$, and $\mathtt{01}$). We call these values the MCL since we cannot distinguish a pair of data points from different clusters if their truncated codes are shorter than the MCL.

Since COOL does not optimize an embedding scheme within the clustering process, the clustering result strongly depends on the embedding used. This means that we have to carefully choose an appropriate embedding for effective clustering. In this paper, we consider the *Gray code* as an embedding scheme for COOL — resulting in an algorithm we call G-COOL. Gray code was originally developed for binary encoding of natural numbers and has become especially important in applications requiring conversion between analog and digital information [17]. From the geometrical point of view, Gray code embedding is the partitioning of each interval into *overlapping* smaller intervals. This enables clusters with arbitrary shapes to be found, which cannot be done with binary embedding. There is theoretical support for clustering by G-COOL as shown in Lemma 2 and Theorem 1. Figure 1 illustrates examples of computing the MCL with binary and Gray code embedding.

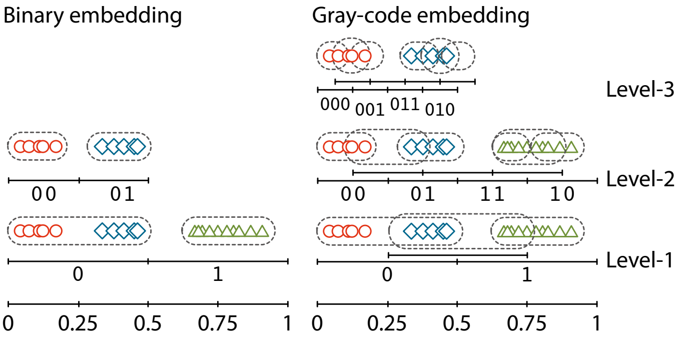


Fig. 1. Examples of computing MCL with binary (left) and Gray code (right) embedding. These one-dimensional data sets are in $[0, 1]$ and partitioned into three clusters, $\bigcirc$, $\diamondsuit$, and $\triangle$. Level means the length of each prefix. With binary embedding, cluster $\triangle$ is separated at level 1, and $\bigcirc$ and $\diamondsuit$ are separated at level 2. They are encoded by `1`, `00`, and `01`, respectively, so the MCL is $1 + 2 + 2 = 5$. With Gray code embedding, the intervals overlap, and adjacent clusters are merged at each level. As a result, $\triangle$ is separated at level 2, and $\bigcirc$ and $\diamondsuit$ are separated at level 3. Their representatives are $\{$`11`, `10`$\}$, $\{$`000`, `001`$\}$, and $\{$`011`, `010`$\}$, respectively, so the MCL is $4 + 6 + 6 = 16$.

The motivation for using Gray code comes from Computable Analysis [31], a well-established mathematical framework for addressing analytical and computational aspects of real numbers in a fully computational manner through representation of real numbers as infinite sequences. Computability for real numbers depends on the embedding method used, and computation makes sense only if the method meets a key mathematical property: “admissibility” (see [31] for its mathematical definition and properties). It is thus natural that the clustering results depends on the embedding method. Gray code has been shown to be admissible [29] whereas binary embedding is not, and this property is a key for embedding that can detect arbitrarily shaped clusters.

This paper is organized as follows: Section 2 gives notation and Section 3 introduces the MCL. Section 4 gives a formal definition of clustering based on the MCL and explains the integration of COOL with the computation of the MCL. In Section 5, we introduce Gray code embedding and analyze G-COOL theoretically. Section 6 describes the experiments, presents the results, and discusses them. Section 7 summarized the key points with reviewing related work.

2 Notation

In the following, $\mathbb{R}^d$ denotes the d-dimensional Euclidean space. A *data point* x is a vector in $\mathbb{R}^d$, and a *dataset* X is a finite set of data points. For a pair of sets X and Y, $X \setminus Y$ means the relative complement of Y in X.

Clustering is the partition of a dataset X into K subsets $C_1, \dots, C_K$, called *clusters*, where $C_i \neq \emptyset$, $C_i \cap C_j = \emptyset$ with $i \neq j$, and $\bigcup_{i \in \{1,\dots,K\}} C_i = X$. Here we say that a set $\mathcal{C} = \{C_1, \dots, C_K\}$ holding the above properties is a *partition* of X and denote the set of all possible partitions by $\mathcal{C}(X)$; *i.e.*, $\mathcal{C}(X) = \{\mathcal{C} \mid \mathcal{C}$ is a partition of $X\}$. For a cluster C, $\#C$ denotes the number of data points in C.

The set of finite and infinite sequences over an alphabet Σ is denoted by Σ^* and Σ^ω, respectively. The length $|w|$ of a finite or an infinite sequence w is the number of positions for symbols other than $\perp$ (the undefinedness character) in w. For example, if $w = \texttt{11}\perp\texttt{100}\perp\perp\perp\dots$, $|w| = 5$. For a set of sequences W, the size of W is defined by $|W| := \sum_{w \in W} |w|$.

An *embedding* of $\mathbb{R}^d$ is an injective function γ from $\mathbb{R}^d$ to Σ^ω. For a pair of infinite sequences p, q, we write $p \leqslant q$ if $p(i) = q(i)$ for all i with $p(i) \neq \perp$, where $p(i)$ denotes the ith position (including 0) of p. This means that q is more specific than p since $\perp$ denotes undefinedness. Moreover, if $w\perp^\omega \leqslant p$ for $w \in \Sigma^*$, we write $w \sqsubset p$ (w is a prefix of p). We define $\uparrow w := \{p \in \text{range}(\gamma) \mid w \sqsubset p\}$ for $w \in \Sigma^*$, and $\uparrow W := \{p \in \text{range}(\gamma) \mid w \sqsubset p \text{ for some } w \in W\}$ for $W \subseteq \Sigma^*$.

3 Minimum Code Length

The minimum code length, or MCL, is used to measure partitions under a fixed embedding γ. We define, for $p \in \text{range}(\gamma)$ and $P \subset \text{range}(\gamma)$,

$$\Phi(p \,|\, P) := \left\{ w \in \Sigma^* \,\middle|\, \begin{array}{l} p \in \uparrow w, \text{ and } P \cap \uparrow v = \emptyset \\ \text{for all } v \text{ with } |v| = |w| \text{ and } p \in \uparrow v \end{array} \right\}.$$

Every element in $\Phi(p \mid P)$ is a prefix of p that discriminates p from P. Trivially, $\Phi(p \mid P) = \emptyset$ if $p \in P$.

The MCL is introduced here in accordance with the above preparations.

Definition 1 (MCL). Given an embedding γ, for a partition $\mathcal{C} = \{C_1, \dots, C_K\}$ of a dataset X, we define

$$\text{MCL}(\mathcal{C}) := \sum_{i \in \{1,\dots,K\}} L_i(\mathcal{C}),$$

where

$$L_i(\mathcal{C}) := \min \left\{ |W| \,\middle|\, \begin{array}{l} \gamma(C_i) \subseteq \uparrow W \text{ and} \\ W \subseteq \displaystyle\bigcup_{x \in C_i} \Phi\left(\gamma(x) \mid \gamma(X \setminus C_i)\right) \end{array} \right\}.$$

Intuitively, this gives the code length of the *maximumly compressed representatives* of a given partition through discretization using fixed embedding γ since the following property holds: For a partition $\mathcal{C}$ of X, if we discretize each data point $x \in X$ into a finite sequence $c(x)$ with γ (*i.e.*, $c(x) \sqsubset \gamma(x)$) such that $\left|\bigcup_{x \in X} c(x)\right| < \text{MCL}(\mathcal{C})$, then there must exist a pair of data points $x, y \in X$ satisfying $\uparrow c(x) \cap \uparrow c(y) \neq \emptyset$ and $x \in C_i, y \in C_j$ with $i \neq j$. Therefore, we cannot discriminate x from y and thus cannot find the partition $\mathcal{C}$ from compressed codes $c(X)$.

Example 1. Suppose we use binary embedding γ_{B}. Assume a one-dimensional dataset $X = \{0.1, 0.2, 0.8, 0.9\}$ and partitions $\mathcal{C}_1 = \{\{0.1, 0.2\}, \{0, 8, 0.9\}\}$ and $\mathcal{C}_2 = \{\{0.1\}, \{0.2, 0.8\}, \{0.9\}\}$. Then, $\mathrm{MCL}(\mathcal{C}_1) = L_1(\mathcal{C}_1) + L_2(\mathcal{C}_1) = 1 + 1 = 2$ since $\gamma_{\mathrm{B}}([0, 0.5]) = {\uparrow}\mathtt{0}$ and $\gamma_{\mathrm{B}}((0.5,\ 1]) = {\uparrow}\mathtt{1}$, and $\mathrm{MCL}(\mathcal{C}_2) = L_1(\mathcal{C}_2) + L_2(\mathcal{C}_2) + L_3(\mathcal{C}_2) = 3 + (3 + 3) + 3 = 12$ because we have $\gamma_{\mathrm{B}}([0, 0.125]) = {\uparrow}\mathtt{000}$, $\gamma_{\mathrm{B}}((0.125, 0.25]) = {\uparrow}\mathtt{001}$, $\gamma_{\mathrm{B}}((0.75, 0.875]) = {\uparrow}\mathtt{110}$, and $\gamma_{\mathrm{B}}((0.875, 1]) = {\uparrow}\mathtt{111}$. Note that $\gamma_{\mathrm{B}}([0, 0.25]) = {\uparrow}\mathtt{00}$, hence 0.1 and 0.2 cannot be discriminated using code $\mathtt{00}$, and that $\gamma_{\mathrm{B}}((0.75, 1]) = {\uparrow}\mathtt{11}$, hence 0.8 and 0.9 cannot be discriminated using $\mathtt{11}$.

The MCL is calculated for $O(nd)$ time complexity by using a radix sort, where n is the size of X (*i.e.*, $n = \#X$), and d is the dimension of X. This is why if the discretized dataset $\{p(0)p(1)\dots p(k-1) \mid p \in \gamma(X)\}$ at level k is sorted in advance, each data point simply needs to be compared with the subsequent one for each dimension, and the MCL is obtained by checking from $k = 1, 2, 3, \dots$.

4 Minimizing MCL and Clustering

We formulate clustering using the MCL as a criterion and describe clustering algorithm COOL, which always finds the globally optimal partition that *minimizes* the MCL.

4.1 Problem Formulation

The clustering problem with the MCL is defined as follows.

Definition 2. *Clustering of a dataset X under the MCL criterion* means finding the globally optimal partition that minimizes the MCL with more than K clusters; that is, finding $\mathcal{C}_{\mathrm{op}}$ such that

$$\mathcal{C}_{\mathrm{op}} \in \operatorname*{argmin}_{\mathcal{C} \in \mathfrak{C}(X)_{\geqslant K}} \mathrm{MCL}(\mathcal{C}),$$

where $\mathfrak{C}(X)_{\geqslant K} = \{\mathcal{C} \in \mathfrak{C}(X) \mid \#\mathcal{C} \geqslant K\}$.

In this framework, we assume that a lower bound on the number of clusters K is given to avoid overgeneralization since, if we search for the optimal partition $\mathcal{C}_{\mathrm{op}}$ in $\mathfrak{C}(X)$ (*i.e.*, all possible partitions) instead of $\mathfrak{C}(X)_{\geqslant K}$, we always have the nonsense result $\mathcal{C}_{\mathrm{op}} = \{X\}$.

4.2 COOL Algorithm

Our COOL algorithm efficiently solves the optimization problem (Definition 2) by integrating the computation of MCL within the clustering step. By contrast, naïve approach that would compare the MCLs of all possible partitions would result in an algorithm with exponential time complexity. The pseudo-code of COOL is shown in Algorithm 1.

COOL is a *level-wise* clustering algorithm that finds the optimal partition $\mathcal{C}_{\mathrm{op}}$ by enumerating level-k partitions ($k = 1, 2, 3, \dots$).

Algorithm 1. COOL algorithm

```
Input: Dataset $X$, lower bound on number of clusters $K$, and noise parameter $N$
Output: Optimal partition $\mathcal{C}_{\mathrm{op}}$ and noise data
Function COOL($X$, $K$, $N$)
  Find partitions $\mathcal{C}^1_{\geqslant N}, \dots, \mathcal{C}^m_{\geqslant N}$ such that $\#\mathcal{C}^{m-1}_{\geqslant N} < K \leqslant \#\mathcal{C}^m_{\geqslant N}$
  ($\mathcal{C}_{\mathrm{op}}$, MCL) ← FINDCLUSTERS($X$, $K$, $\{\mathcal{C}^1_{\geqslant N}, \dots, \mathcal{C}^m_{\geqslant N}\}$)
  return ($\mathcal{C}_{\mathrm{op}}$, $X \setminus \bigcup \mathcal{C}_{\mathrm{op}}$)
Function FINDCLUSTERS($X$, $K$, $\{\mathcal{C}^l, \dots, \mathcal{C}^m\}$)
  if $K = 1$ then
    return ($\mathcal{C}^l$, $|W|$), where $\gamma(\bigcup \mathcal{C}^l) \subseteq {\uparrow}W$ and $|w| = l$ for all $w \in W$
  end if
  Find $k$ such that $\#\mathcal{C}^{k-1} < K \leqslant \#\mathcal{C}^k$
  $\mathcal{C}_{\mathrm{op}} \leftarrow \mathcal{C}^k$
  MCL ← MCL($\mathcal{C}^k$)
  for each $C$ in $\mathcal{C}^l \cup \dots \cup \mathcal{C}^k$
    $L \leftarrow \min\{|W| \mid \gamma(C) \subseteq {\uparrow}W$ and $|w| = j$ for all $w \in W\}$, where $C \in \mathcal{C}^j$
    ($\mathcal{C}$, $L'$) ← FINDCLUSTERS($X \setminus C$, $K - 1$, $\{\mathcal{C}^j, \dots, \mathcal{C}^k\}$)
    if $L + L' <$ MCL then
      $\mathcal{C}_{\mathrm{op}} \leftarrow C \cup \mathcal{C}$
      MCL ← $L + L'$
    end if
  end for
  return ($\mathcal{C}_{\mathrm{op}}$, MCL)
```

Definition 3 (Level-*k* partition). For a dataset X and an embedding γ, the *level-k partition* $\mathcal{C}^k$ is defined as follows: Every pair of data points $x, y \in X$ are contained in the same cluster if and only if $v = w$ for some $v \sqsubset \gamma(x)$ and $w \sqsubset \gamma(y)$ with $|v| = |w| = k$.

This means that if $x, y \in X$ are in the same cluster, there exists a *chain* of data points $z_1, z_2, \dots, z_m$ ($m \geqslant 2$) such that, for all $i \in \{1, 2, \dots, m-1\}$, $z_1 = x$, $z_m = y$, and $w_i = w_{i+1}$ for some $w_i \sqsubset \gamma(z_i)$ and $w_{i+1} \sqsubset \gamma(z_{i+1})$ with $|w_i| = |w_{i+1}| = k$. Obviously, the level-k partition is determined uniquely. The time complexity of finding the level-k partition is $O(nd)$, where n and d are the size and dimension of the dataset, respectively, since, if the discretized dataset $\{p(0)p(1)\dots p(k-1) \mid p \in \gamma(X)\}$ at level k is initially sorted using a radix sort, clusters are constructed by comparing each data point to the next data point for each dimension.

The most important feature of the level-k partition is that the optimal partition $\mathcal{C}_{\mathrm{op}}$ in Definition 2 is obtained by searching for only clusters contained in the level-k partition.

Lemma 1. *For every cluster $C \in \mathcal{C}_{\mathrm{op}}$, C is contained in some level-k partition, that is, $C \in \mathcal{C}^k$ for some $k \in \mathbb{N}$.*

Proof. Let $\mathcal{C}$ be a partition such that, for every $C \in \mathcal{C}$, $C \in \mathcal{C}^k$ for some k, and a pair of clusters $C, C' \in \mathcal{C}$ is fixed. Then, from the definition of the level-k partition, the following condition holds: For all pairs of clusters D, D' such that $D \cup D' = C \cup C'$ and $D \cap D' = \emptyset$, we have $\mathrm{MCL}(\mathcal{C}) \leqslant \mathrm{MCL}(\mathcal{C}')$, where $\mathcal{C}' = (\mathcal{C} \setminus \{C, C'\}) \cup \{D, D'\}$. Therefore, for the optimal partition $\mathcal{C}_{\mathrm{op}}$, $C \in \mathcal{C}^k$ with $k \in \mathbb{N}$ for all $C \in \mathcal{C}_{\mathrm{op}}$. □

The level-k partition has a *hierarchical* structure: For each cluster $C \in \mathcal{C}^k$, there must exist a set of clusters $\mathcal{D} \subseteq \mathcal{C}^{k+1}$ such that $\bigcup \mathcal{D} = C$. Thus, COOL works through divisive hierarchical clustering. The MCL of the level-k partition used in line 6 of the function FindClusters in Algorithm 1 can thus be easily calculated: Let $\mathcal{C}^k$ be a set of clusters $\{C_1, \dots, C_K\}$. For each C_i and for the minimum level l such that $C_i \in \mathcal{C}^l$,

$$L_i(\mathcal{C}^k) = \min\{|W| \mid \gamma(C_i) \subseteq {\uparrow}W \text{ and } |w| = l \text{ for all } w \in W\}$$

holds. This means that we can obtain the MCL of the level-k partition by checking only sequences with length l.

Next we show that COOL can solve the optimization problem in Definition 2.

Proposition 1. *The* COOL *algorithm* (*Algorithm 1*) *always outputs the globally optimal partition* $\mathcal{C}_{\mathrm{op}}$.

Proof. Let $\#\mathcal{C}_{\mathrm{op}} = K$. Then there must exist $k \in \mathbb{N}$ such that $K \leqslant \#\mathcal{C}^k$ and $\#\mathcal{C}^{k'} < K$ for all $k' < k$ since the number of clusters in the level-k partition $\#\mathcal{C}^k$ increases monotonically with respect to increase of k. Fix a cluster $C \in \mathcal{C}_{\mathrm{op}}$, and let $\mathcal{C}'_{\mathrm{op}}$ be the optimal partition for the dataset $X \setminus C$. Then we can easily check that $\mathcal{C}'_{\mathrm{op}} \cup \{C\}$ coincides with $\mathcal{C}_{\mathrm{op}}$. Moreover, from Lemma 1 and the definition of the level-k partition, for all $C \in \mathcal{C}_{\mathrm{op}}$, $C \in \mathcal{C}^l$ for some $l \in \{1, \dots, k\}$. Thus, COOL finds the optimal partition $\mathcal{C}_{\mathrm{op}}$ by recursive computing in Algorithm 1 (lines 4 - 7) with fixing each cluster in $\mathcal{C}^1 \cup \dots \cup \mathcal{C}^k$. □

COOL can find the globally optimal partition $\mathcal{C}_{\mathrm{op}}$ efficiently, and its time complexity is $O(nd)$ and $O(nd + K!)$ in the best and worst cases, respectively, since finding m partitions in the first line of the function Cool takes $O(nd)$, and the function FindClusters takes $O(K!)$ in the worst case. Usually, $K \ll n$ holds, so complexity becomes $O(nd)$.

Furthermore, noise is directly removed by COOL using a lower bound on the size of each cluster N, which we call the *noise parameter*. For a partition $\mathcal{C}$, we denote the set $\{C \in \mathcal{C} \mid \#C \geqslant N\}$ by $\mathcal{C}_{\geqslant N}$. For example, let a dataset $X = \{0.1, 0.4, 0.5, 0.6, 0.9\}$ and $\mathcal{C} = \{\{0.1\}, \{0.4, 0.5, 0.6\}, \{0.9\}\}$. Then, $\mathcal{C}_{\geqslant 2} = \{\{0.4, 0.5, 0.6\}\}$, and two data points, 0.1 and 0.9, are detected as noise.

5 G-COOL: COOL with Gray Code

We use Gray code embedding for COOL, and show its powerful clustering ability by theoretical analysis. We call COOL with Gray code embedding G-COOL.

5.1 Gray Code Embedding

Gray code embedding is illustrated in Figure 2. Its rich mathematical properties are described elsewhere [29]. Gray code was originally simply binary encoding of natural numbers, as mentioned in introduction. For example, natural numbers $1, 2, \dots, 8$ are represented in Gray code as `000`, `001`, `011`, `010`, `110`, `111`, `101`, `100`, whereas, in binary code, they are represented as `000`, `001`, `010`, `011`, `100`, `101`, `110`, `111`. The importance of Gray code is that only one bit differs between one code and its successor, that is, the Hamming distance between them is always one. Here $\mathcal{I}$ denotes the unit interval $[0,1] \times \cdots \times [0,1] \subset \mathbb{R}^d$, and $\Sigma^\omega_{\perp,d}$ denotes the set of infinite sequences for which, in each sequence, at most d positions are $\perp$. For example, if $\Sigma = \{\mathtt{0}, \mathtt{1}\}$ and $d = 2$, then $\mathtt{0}\perp\mathtt{100}\cdots \in \Sigma^\omega_{\perp,d}$, $\perp\perp\mathtt{110}\cdots \in \Sigma^\omega_{\perp,d}$, and $\mathtt{0}\perp\mathtt{1}\perp\perp\mathtt{0}\dots \notin \Sigma^\omega_{\perp,d}$. In the following, we consider only real vectors in $\mathcal{I}$.

Definition 4 (Gray code embedding). (*One-dimensional*) *Gray code embedding* is an injective function, $\gamma_{\mathrm{G}} : \mathcal{I} \to \Sigma^\omega_{\perp,d}$ ($d = 1$), that maps $x \in \mathcal{I}$ to an infinite sequence $p(0)p(1)p(2)\dots$: For each i, $p(i) := \mathtt{1}$ if

$$2^{-i}m - 2^{-(i+1)} < x < 2^{-i}m + 2^{-(i+1)}$$

holds for an odd number m, $p(i) := \mathtt{0}$ if the same holds for an even number m, and $p(i) := \perp$ if $x = 2^{-i}m - 2^{-(i+1)}$ for some integer m.

Moreover, by using the *wrapping function*

$$\varphi(p_1, \dots, p_d) := p_1(0)\dots p_d(0)p_1(1)\dots p_d(1)p_1(2)\dots p_d(2)\dots,$$

we can define *d-dimensional Gray code embedding* $\gamma^d_{\mathrm{G}} : \mathcal{I} \to \Sigma^\omega_{\perp,d}$ as

$$\gamma^d_{\mathrm{G}}(x_1, \dots, x_d) := \varphi(\gamma_{\mathrm{G}}(x_1), \dots, \gamma_{\mathrm{G}}(x_d)).$$

We abbreviate d of γ^d_{G} if it is understood from the context.

Example 2. For one-dimensional data points $x = 0.2$, $y = 0.5$, and $z = 0.7$, we have $\gamma_{\mathrm{G}}(x) = \mathtt{0010}\dots$, $\gamma_{\mathrm{G}}(y) = \perp\mathtt{100}\dots$, and $\gamma_{\mathrm{G}}(z) = \mathtt{1110}\dots$ with Gray code embedding, while $\gamma_{\mathrm{B}}(x) = \mathtt{0001}\dots$, $\gamma_{\mathrm{B}}(y) = \mathtt{0111}\dots$, and $\gamma_{\mathrm{B}}(z) = \mathtt{1011}\dots$ with binary embedding. For a two-dimensional data point (x, y), we have $\gamma_{\mathrm{G}}(x, y) = \mathtt{0}\perp\mathtt{011000}\dots$, and for a three-dimensional data point (x, y, z), $\gamma_{\mathrm{G}}(x, y, z) = \mathtt{0}\perp\mathtt{1}$ $\mathtt{011101000}\dots$ with Gray code embedding.

5.2 Theoretical Analysis of G-COOL

Here we show that G-COOL achieves internal cohesion and external isolation without any distance calculation or data distribution. In the following, we measure the distance between $x, y \in \mathbb{R}^d$ by using the L_∞ metric, where the distance is defined by

$$d_\infty(x, y) := \max_{i \in \{1,\dots,d\}} |x_i - y_i|\,.$$

Lemma 2. *For the level-k partition $\mathcal{C}^k$ of a dataset X with Gray code embedding γ_{G}, two data points $x, y \in X$ are in the same cluster if $d_\infty(x, y) < 2^{-(k+1)}$ and are not in the same cluster only if $d_\infty(x, y) \geqslant 2^{-(k+1)}$.*

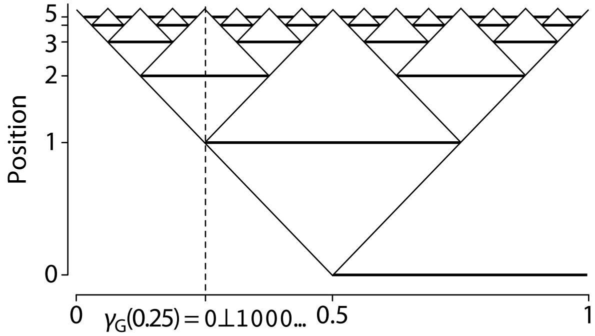


Fig. 2. Gray code embedding γ_{G}. Position i is `1` if it is on the line, $\perp$ if on the end point, and `0` otherwise. Diagonal lines are auxiliary lines. For example, if $p = \gamma_{\mathrm{G}}(0.25)$, $p = \mathtt{0}\perp\mathtt{1000}\ldots$ because position 0 is not on the line, 1 is on the end point, 2 is on the line, and every $i \geqslant 3$ is not on the line.

Proof. From the definition of Gray code embedding, if $d_\infty(x, y) < 2^{-(k+1)}$ for $x, y \in X$, there must exist a finite sequence w with $|w| = k$ such that $w \sqsubset \gamma_{\mathrm{G}}(x)$ and $w \sqsubset \gamma_{\mathrm{G}}(y)$. Thus, x and y are in the same cluster in the level-k partition $\mathcal{C}^k$. Moreover, this means that, if x and y are in the different clusters in $\mathcal{C}^k$, $d_\infty(x, y) \geqslant 2^{-(k+1)}$. □

Informally, the *redundancy* of Gray code embedding enables the powerful property described in the above lemma, that is, for an infinite sequence $p = \gamma_{\mathrm{G}}(x)$, there may be two prefixes, $v_1 \sqsubset p$ and $v_2 \sqsubset p$ with $|v| = |w|$.

Example 3. Let us consider the situation illustrated in Figure 3, where we have five one-dimensional data points: $x_{\mathrm{a}} = 0.14$, $x_{\mathrm{b}} = 0.48$, $x_{\mathrm{c}} = 0.51$, $x_{\mathrm{d}} = 0.73$, and $x_{\mathrm{e}} = 0.77$. In binary embedding, the unit interval $\mathcal{I} = [0, 1]$ is divided into two intervals $[0, 0.5]$ and $[0.5, 1]$ at level-1, while it is divided into three intervals $[0, 0.5]$, $[0.25, 0.75]$, and $[0.5, 1]$ in Gray code embedding. Thus, there are three overlapping clusters $\{x_{\mathrm{a}}, x_{\mathrm{b}}\}$ (encoded as `0`), $\{x_{\mathrm{b}}, x_{\mathrm{c}}, x_{\mathrm{d}}\}$ (encoded as $\perp$`1`), and $\{x_{\mathrm{c}}, x_{\mathrm{d}}, x_{\mathrm{e}}\}$ (encoded as `1`). Actually, there is only one cluster $\{x_{\mathrm{a}}, x_{\mathrm{b}}, x_{\mathrm{c}}, x_{\mathrm{d}}, x_{\mathrm{e}}\}$ since they are merged. At level-2, we have four clusters with binary embedding although some data points such as x_{b} and x_{c} are close. On the other hand, we have two *natural* clusters $\{x_{\mathrm{a}}\}$ and $\{x_{\mathrm{b}}, x_{\mathrm{c}}, x_{\mathrm{d}}, x_{\mathrm{e}}\}$ with Gray code embedding.

Intuitively, this lemma theoretically supports the claim that G-COOL finds *natural* clusters. For a data point $x \in X$, we say that a data point $y \in X$ is the *nearest neighbor* of x if $y \in \mathrm{argmin}_{x' \in X} d_\infty(x, x')$.

Theorem 1 (main theorem). *The optimal partition $\mathcal{C}_{\mathrm{op}}$ of a dataset X generated by* G-COOL *has the following property: For every data point $x \in C$ with $C \in \mathcal{C}_{\mathrm{op}}$ and $\#C \geqslant 2$, its nearest neighbor $y \in C$.*

Proof. From Lemma 1, every cluster $C \in \mathcal{C}_{\mathrm{op}}$ is contained in $\mathcal{C}^k$ for some k. Thus, from Lemma 2, trivially, any $x \in C$ with $C \in \mathcal{C}_{\mathrm{op}}$, $\#C = 1$ or its nearest neighbor $y \in C$. □

id	Value	Level 1		Level 2	
		Binary	Gray	Binary	Gray
a	0.14	0	0	00	00
b	0.48	0	0, ⊥1	01	01, ⊥10
c	0.51	1	1, ⊥1	10	11, ⊥10
d	0.73	1	1, ⊥1	10	11, 1⊥1
e	0.77	1	1	11	10, 1⊥1

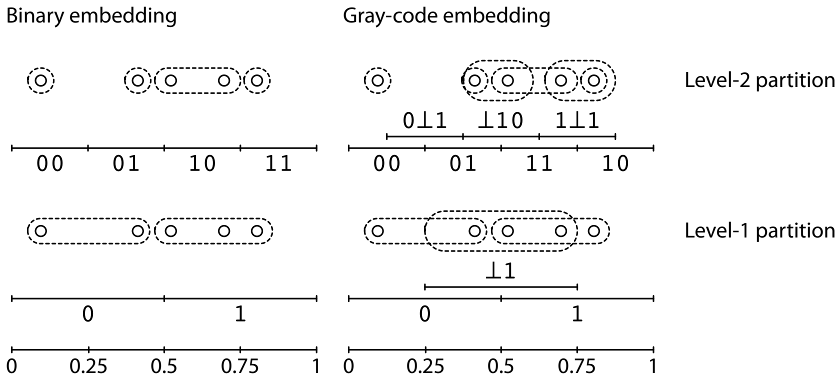


Fig. 3. Examples of level-1 and 2 partitions with binary and Gray code embedding

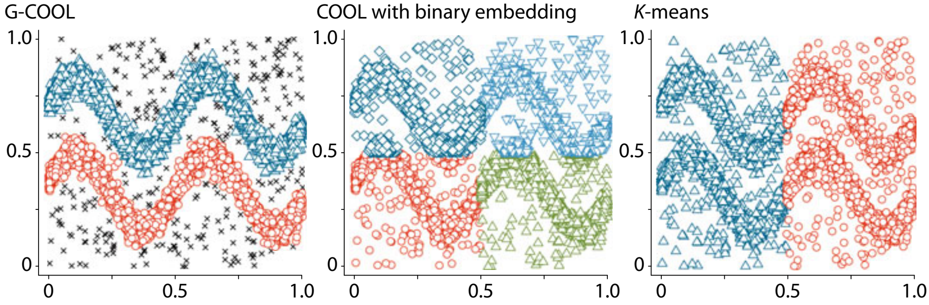


Fig. 4. Clustering results for G-COOL and COOL with binary embedding ($K = 2, N = 50$) and K-means ($K = 2$). Dataset size is 10,500 (where 500 points are noise). G-COOL detects two natural clusters and noise. The other two methods cannot find such clusters.

This property of Gray code (Lemma 2) enables clusters with the condition in Theorem 1 to be quickly found, whereas the naïve solution results in more than $O(n^2)$. Figure 4 illustrates the results of G-COOL for a two-dimensional dataset for which K-means could not find natural clusters. We can see that COOL with binary embedding also failed to find such clusters.

6 Experiments

We analyze G-COOL empirically and evaluate the effectiveness of G-COOL and the proposed measure, MCL. We use low-dimensional synthetic and real datasets, which are common in spatial clustering setting.

6.1 Methods

Environment. G-COOL was implemented in R version 2.12.2 [24], and all experiments were performed in R. We used Mac OS X version 10.6.5 with two 2.26-GHz Quad-Core Intel Xeon CPUs and 12 GB of memory.

Datasets. The synthetic datasets were used to evaluate robustness against the number of clusters, the size of the datasets, and noise existence. They were randomly generated using the R `clusterGeneration` package [23], and the parameters were set as follows: $sepVal = 0.34$, $numNonNoisy = 2$, $numNoisy = 1$, $numOutlier = 500$, and $rangeN = c(1000, 2000)$. We generated 20 datasets to obtain the mean and s.e.m. (standard error of the mean). The size of each cluster was around 1,500, so the size of the datasets varied from ∼3,000 to ∼10,500. Each dataset was three-dimensional, where one dimension was composed of noise and about 500 data point were added to each dimension as noise.

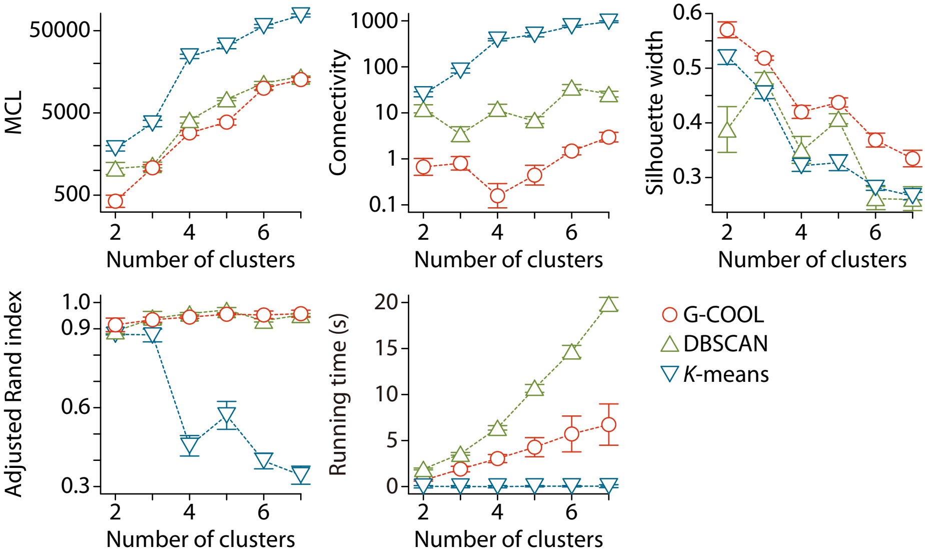


Fig. 5. Experimental results for synthetic datasets. MCL and connectivity should be minimized, and Silhouette width and adjusted Rand index should be maximized. Data show mean ± s.e.m.

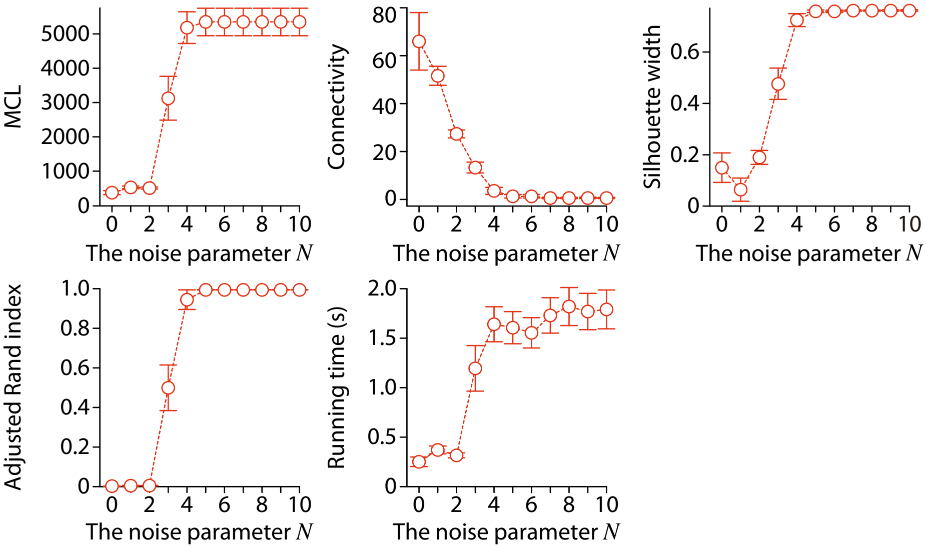


Fig. 6. Clustering speed and quality for G-COOL and synthetic datasets. MCL and connectivity should be minimized, and Silhouette width and adjusted Rand index should be maximized. G-COOL shows robust performance if the noise parameter N is large enough. Data show mean ± s.e.m.

Five real datasets were collected from the Earth-as-Art website[1], which contains geospatial satellite images (see Table 1 and Figure 7). Similar datasets were used in experiments with the state-of-the-art spatial clustering method [5]. Each image was pre-processed using ImageJ software [25]; they were reduced to 200 × 200 pixels and translated into binary images.

With G-COOL, each dataset was translated using min-max normalization [10] so that the dataset was in the unit interval $\mathcal{I}$, where each value x of ith dimension X_i of a dataset X was mapped to $x' = (x - \min X_i)/(\max X_i - \min X_i)$ and the runtime for the translation was included in the G-COOL running time.

Control Methods. As control methods, we used K-means and DBSCAN because K-means is the standard clustering algorithm and DBSCAN is a typical method for finding arbitrarily shaped clusters, and their source codes are publicly available. DBSCAN was executed using the R `fpc` package. We tuned the parameters of DBSCAN to obtain the best results.

Evaluation. With the synthetic datasets, performance was evaluated using internal and external measures. As internal measures, we used the MCL (with Gray code), the connectivity (takes values in $[0, \infty]$, to be minimized) [11], and the Silhouette width (takes values in $[-1, 1]$, to be maximized) [26]. As an external measure, we used the adjusted Rand index (takes values in $[-1, 1]$, to be maximized) [13], which takes into account the

[1] `http://eros.usgs.gov/imagegallery/`

Table 1. Running time (in seconds) and MCL for real datasets. In table, n and K denote the number of data points and clusters, respectively. Clustering results are shown in Figure 7.

Name	n	K	Running time (s)		MCL	
			G-COOL	K-means	G-COOL	K-means
Delta	20748	4	1.158	0.012	4010	4922
Dragon	29826	2	0.595	0.026	3906	7166
Europe	17380	6	2.404	0.041	2320	12210
Norway	22771	5	0.746	0.026	1820	6114
Ganges	18019	6	0.595	0.026	2320	12526

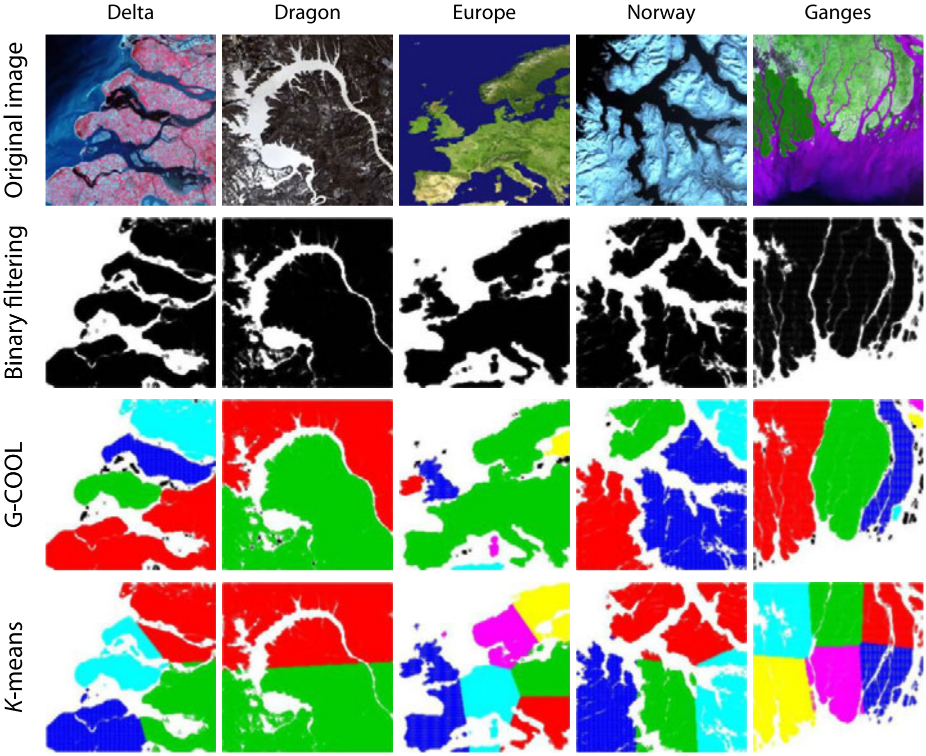


Fig. 7. Results for real datasets obtained from satellite images by G-COOL and K-means. G-COOL finds all natural clusters whereas K-means does not.

ground truth and is popular in the clustering literature. The measures were calculated using the R `clValid` [2], `cluster` [22], and `clues` [3] packages, respectively. For the real datasets, we used the MCL and simply show scatter plots of the results since we had no information on the ground truth.

6.2 Results and Discussion

The results obtained with the synthetic datasets (Figure 5) show that the quality of clusters obtained with G-COOL is significantly higher for three of the four quality measures (determined by paired t-test) and is competitive for the other one (adjusted Rand index). Moreover, it is faster than DBSCAN. These results show that the MCL works reasonably well as a measure of cluster quality compared to existing ones.

Note that we need to input only the lower bounds for the number and size of the clusters in G-COOL, whereas we have to tune the parameters carefully in DBSCAN and other shape-based (spatial) clustering algorithms. Therefore, G-COOL is more efficient and effective than existing clustering algorithms. Moreover, as shown in Figure 6, cluster quality is stable with respect to the noise parameter N (*i.e.*, lower bound on cluster size) even if the dataset contains noise, when N is large enough. If N is too small, then each noise is detected as a cluster. Thus, when clustering using G-COOL, all we have to do is set the parameter large enough, meaning that G-COOL is equally useful as K-means.

For all the real datasets, G-COOL finds natural clusters (N was set as 50 for all the datasets), as shown in Figure 7, whereas K-means results in inferior clustering quality (we did not perform DBSCAN since it takes too much time and needs manual tuning of the input parameters). Moreover, MCL of clustering results for G-COOL is much smaller than those for K-means (Table 1). These results show that G-COOL is robust and that it can find arbitrarily shaped clusters without careful tuning of the input parameters.

7 Conclusion

We have proposed an internal measure, the minimum code length (MCL), to evaluate the results of clustering and presented a clustering algorithm COOL that always finds the globally optimal partition; *i.e.*, clusters that have the minimum MCL. Intuitively, COOL produces the maximally compressed clusters using a fixed encoding scheme and does not take optimization of encoding into account. Moreover, Gray code is used for the encoding, resulting in an algorithm called G-COOL. Theoretically and empirically, G-COOL has been shown to be noise tolerant and to have the ability to find arbitrarily shaped clusters efficiently. The result is an effective solution to two essential problems, how to measure the goodness of clustering results and how to find good clusters.

Many types of *shape-based clustering*, or *spatial clustering*, methods have been proposed for finding arbitrarily shaped clusters, including partitional algorithms [4,5], the mass-based clustering algorithm [28], density-based clustering algorithms (*e.g.*, DBSCAN [7] and DENCLUE [12]), agglomerative hierarchical clustering algorithms (*e.g.*, CURE [8], CHAMELEON [15]), and grid-based algorithms (*e.g.*, STING [30] and Wave Cluster [27]). However, most of them are not practical. Their clustering results are sensitive to the input parameters, which have to be tuned manually, so they work well only if all parameters are tuned appropriately by the user. As a result, these methods are not well suited for users who are not specialized in machine learning. Furthermore, most of them are not scalable: their time complexity is quadratic or cubic with respect to data size. Compared to these methods, G-COOL is robust to the input parameters and always finds the globally optimal clusters under the MCL criterion. Moreover, G-COOL is usually faster than most of these methods since the time complexity is linear with respect to data size.

Many cluster validity methods have been proposed for quantitative evaluation of clustering results [11]. These measures are usually divided into two categories: internal (*e.g.*, connectivity and Silhouette width) and external (*e.g.*, F-measure and Rand index). The internal measures are intrinsic to actual clustering while the external measures need information that may not be available in an actual situation. Our proposed measure, MCL, can be categorized as an internal measure. Its effectiveness has been shown experimentally (see Section 6).

G-COOL's results are robust to changes in the input parameters, and does not assume a data distribution and does not need a distance calculation. Thus, it can be effectively applied to other machine learning tasks, such as anomaly detection. Theoretical analysis of relationship between admissibility of encoding schemes in computing real numbers and the ability of clustering to detect arbitrarily shaped clusters is necessary future work.

Acknowledgments. We would like to thank Marco Cuturi for his helpful comments. This work was partly supported by Grant-in-Aid for Scientific Research (A) 22240010 and for JSPS Fellows 22·5714.

References

1. Berkhin, P.: A survey of clustering data mining techniques. Grouping Multidimensional Data, 25–71 (2006)
2. Brock, G., Pihur, V., Datta, S., Datta, S.: clValid: An R package for cluster validation. Journal of Statistical Software 25(4), 1–22 (2008)
3. Chang, F., Qiu, W., Zamar, R.H., Lazarus, R., Wang, X.: clues: An R package for nonparametric clustering based on local shrinking. Journal of Statistical Software 33(4), 1–16 (2010), http://www.jstatsoft.org/v33/i04/
4. Chaoji, V., Hasan, M.A., Salem, S., Zaki, M.J.: SPARCL: An effective and efficient algorithm for mining arbitrary shape-based clusters. Knowledge and Information Systems 21(2), 201–229 (2009)
5. Chaoji, V., Li, G., Yildirim, H., Zaki, M.J.: ABACUS: Mining arbitrary shaped clusters from large datasets based on backbone identification. In: Proceedings of 2011 SIAM International Conference on Data Mining, pp. 295–306 (2011)
6. Cilibrasi, R., Vitányi, P.M.B.: Clustering by compression. IEEE Transactions on Information Theory 51(4), 1523–1545 (2005)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, vol. 96, pp. 226–231 (1996)
8. Guha, S., Rastogi, R., Shim, K.: CURE: An efficient clustering algorithm for large databases. Information Systems 26(1), 35–58 (1998)
9. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems 17(2), 107–145 (2001)
10. Han, J., Kamber, M.: Data Mining, 2nd edn. Morgan Kaufmann, San Francisco (2006)
11. Handl, J., Knowles, J., Kell, D.B.: Computational cluster validation in post-genomic data analysis. Bioinformatics 21(15), 3201 (2005)

12. Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, pp. 58–65 (1998)
13. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification 2(1), 193–218 (1985)
14. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys 31(3), 264–323 (1999)
15. Karypis, G., Eui-Hong, H., Kumar, V.: CHAMELEON: Hierarchical clustering using dynamic modeling. Computer 32(8), 68–75 (1999)
16. Keogh, E., Lonardi, S., Ratanamahatana, C., Wei, L., Lee, S.H., Handley, J.: Compression-based data mining of sequential data. Data Mining and Knowledge Discovery 14, 99–129 (2007)
17. Knuth, D.E.: The Art of Computer Programming. Fascicle 2: Generating All Tuples and Permutations, vol. 4. Addison-Wesley Professional, Reading (2005)
18. Kontkanen, P., Myllymäki, P.: An empirical comparison of NML clustering algorithms. In: Proceedings of Information Theory and Statistical Learning, pp. 125–131 (2008)
19. Kontkanen, P., Myllymäki, P., Buntine, W., Rissanen, J., Tirri, H.: An MDL framework for data clustering. In: Grünwald, P., Myung, I.J., Pitt, M. (eds.) Advances in Minimum Description Length: Theory and Applications. MIT Press, Cambridge (2005)
20. Li, M., Badger, J.H., Chen, X., Kwong, S., Kearney, P., Zhang, H.: An information-based sequence distance and its application to whole mitochondrial genome phylogeny. Bioinformatics 17(2), 149–154 (2001)
21. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297 (1967)
22. Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M.: Cluster analysis basics and extensions (2005)
23. Qiu, W., Joe, H.: Generation of random clusters with specified degree of separation. Journal of Classification 23, 315–334 (2006)
24. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing (2011), http://www.R-project.org
25. Rasband, W.S.: ImageJ. U. S. National Institutes of Health, Bethesda, Maryland, USA (1997–2011), http://imagej.nih.gov/ij/
26. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20, 53–65 (1987)
27. Sheikholeslami, G., Chatterjee, S., Zhang, A.: WaveCluster: A multi-resolution clustering approach for very large spatial databases. In: Proceedings of the 24th International Conference on Very Large Data Bases, pp. 428–439 (1998)
28. Ting, K.M., Wells, J.R.: Multi-dimensional mass estimation and mass-based clustering. In: Proceedings of 10th IEEE International Conference on Data Mining, pp. 511–520 (2010)
29. Tsuiki, H.: Real number computation through Gray code embedding. Theoretical Computer Science 284(2), 467–485 (2002)
30. Wang, W., Yang, J., Muntz, R.: STING: A statistical information grid approach to spatial data mining. In: Proceedings of the 23rd International Conference on Very Large Data Bases, pp. 186–195 (1997)
31. Weihrauch, K.: Computable Analysis: An Introduction. Springer, Heidelberg (2000)

Learning to Infer Social Ties in Large Networks*

Wenbin Tang, Honglei Zhuang, and Jie Tang

Department of Computer Science and Technology, Tsinghua University
{tangwb06,honglei.zhuang}@gmail.com, jietang@tsinghua.edu.cn

Abstract. In online social networks, most relationships are lack of meaning labels (e.g., "colleague" and "intimate friends"), simply because users do not take the time to label them. An interesting question is: can we automatically infer the type of social relationships in a large network? what are the fundamental factors that imply the type of social relationships? In this work, we formalize the problem of social relationship learning into a semi-supervised framework, and propose a Partially-labeled Pairwise Factor Graph Model (PLP-FGM) for learning to infer the type of social ties. We tested the model on three different genres of data sets: Publication, Email and Mobile. Experimental results demonstrate that the proposed PLP-FGM model can accurately infer 92.7% of advisor-advisee relationships from the coauthor network (Publication), 88.0% of manager-subordinate relationships from the email network (Email), and 83.1% of the friendships from the mobile network (Mobile). Finally, we develop a distributed learning algorithm to scale up the model to real large networks.

1 Introduction

With the success of many large-scale online social networks, such as Facebook, MySpace, and Twitter, and the rapid growth of mobile social networks such as FourSquare, online social network has become a bridge between our real daily life and the virtual web space. Facebook, one of the largest social networks, has more than 600 million active users in Jan 2011; Foursquare, a location-based mobile social network, has attracted 6 million registered users by the end of 2010. Just to mention a few, there is little doubt that most of our friends are online now. Considerable research has been conducted on social network analysis [1,7,18,21], dynamic evolution analysis [13], social influence analysis [5,12,23], and social behavior analysis [20,22]. However, most of these works ignore one important fact that makes the online social networks very different from the physical social networks, i.e., our physical social networks are colorful ("family members", "colleagues", and "classmates") but the online social networks are still black-and-white: the users merely do not take the time to label the relationships. Indeed, statistics show that only 16% of mobile phone users in Europe

* The work is supported by the Natural Science Foundation of China (No. 61073073, No. 60973102), Chinese National Key Foundation Research (No. 60933013, No.61035004).

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 381–397, 2011.

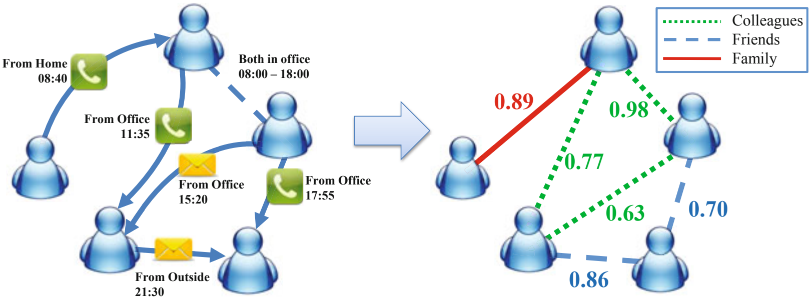


Fig. 1. An example of relationship mining in mobile communication network. The left figure is the input of our problem, and the right figure is the objective of the relationship mining task.

have created custom contact groups [20,10] and less than 23% connections on LinkedIn have been labeled. Identification of the type of social relationships can benefit many applications. For example, if we could have extracted friendships between users from the mobile communication network, we can leverage the friendships for a "word-of-mouth" promotion of a new product [12].

In this work, we investigate to what extent social relationships can be inferred from the online social networks: E.g., given users' behavior history and interactions between users, can we estimate how likely they are to be family members? There exist a few related studies. For example, Diehl et al. [4] try to identify the relationships by learning a ranking function. Wang et al. [26] propose an unsupervised algorithm for mining the advisor-advisee relationships from the publication network. However, both algorithms focus on a specific domain (Email network in [4] and Publication network in [26]) and are not easy to extend to other domains. It is well recognized that the type of users' relationships in a social network can be implied by various complex and subtle factors [9,14]. One challenging question is: can we design a unified model so that it can be easily applied to different domains?

Motivating Examples. To illustrate the problem, Figure 1 gives an example of relationship mining in mobile calling network. The left figure is the input of our problem: a mobile social network, which consists of users, calls and messages between users, and users' location logs, etc. Our objective is to infer the type of the relationships in the network. In the right figure, the users who are family members are connected with a red-colored line, friends are connected with a blue-colored dash line, and colleagues are connected with a green-colored dotted line. The probability associated with each relationship represents our confidence on the detected relationship types.

Thus, the problem becomes how to design a flexible model for effectively and efficiently mining relationship types in different networks. This problem is non-trivial and poses a set of unique challenges. First, what are the underlying factors that may determine a specific type of social relationship. Second, the input social network is partially labeled. We may have some labeled relationships, but most of the relationships are unknown. To learn a high-quality predictive model, we should not only consider the knowledge provided by the labeled relationships, but also leverage the unlabeled network information. Finally, real social networks are getting bigger with thousands even millions of nodes. It is important to develop a method that can scale well to real large networks.

Contributions. In this paper, we try to conduct a systematic investigation of the problem of inferring social relationship types in large networks with the following contributions:

- We formally formulate the problem of inferring social relationship in large networks, and propose a partially-labeled pairwise factor graph model (PLP-FGM).
- We present a distributed implementation of the learning algorithm based on MPI (Message-Passing Interface) to scale up to large networks.
- We conduct experiments on three different data sets: Publication, Email, Mobile network. Experimental results show that the proposed PLP-FGM model can be applied to the different scenarios and clearly achieves better performance than several alternative models.

The rest of paper is organized as follows. Section 2 formally formulates the problem; Section 3 explains the PLP-FGM model; Section 4 gives experimental results; Finally, Section 5 discusses related work and Section 6 concludes.

2 Problem Definition

In this section, we first give several necessary definitions and then present the problem formulation.

A social network can be represented as $G = (V, E)$, where V is the set of $|V| = N$ users and $E \subset V \times V$ is the set of $|E| = M$ relationships between users. The objective of our work is to learn a model that can effectively infer the type of social relationships between two users. To begin with, let us first give a formal definition of the output of the problem, namely "relationship semantics".

Definition 1. ***Relationship semantics:*** *Relationship semantics is a triple* (e_{ij}, r_{ij}, p_{ij})*, where* $e_{ij} \in E$ *is a social relationship,* $r_{ij} \in \mathcal{Y}$ *is a label associated with the relationship, and* p_{ij} *is the probability (confidence) obtained by an algorithm for inferring relationship type.*

Social relationships might be undirected in some networks (e.g., the friendship discovered from the mobile calling network) or directed in other networks (e.g., the advisor-advisee relationship in the publication network). To be consistent, we define all social relationships as directed relationships. In addition, relationships may be static (e.g., the family-member relationship) or change over time (e.g., colleague relationship). In this work, we focus on static relationships, and leave the dynamic case to our future work.

To infer relationship semantics, we could consider different factors such as user-specific information, link-specific information, and global constraints. For example, to discover advisor-advisee relationships from a publication network, we can consider how many papers were coauthored by two authors; how many papers in total an author has published; when the first paper was published by each author. Besides, there may already exist some labeled relationships. Formally, we can define the input of our problem, a partially labeled network.

Definition 2. ***Partially labeled network:*** *A partially labeled network is an augmented social network denoted as* $G = (V, E^L, E^U, R^L, \mathbf{W})$*, where* E^L *is a set of labeled relationships and* E^U *is a set of unlabeled relationships with* $E^L \cup E^U = E$*;* R^L *is a set of labels corresponding to the relationships in* E^L*;* W *is an attribute matrix associated with users in* V *where each row corresponds to a user, each column an attribute, and an element* w_{ij} *denotes the value of the* j^{th} *attribute of user* v_i*.*

Based on the above concepts, we can define the problem of inferring social relationships. Given a partially labeled network, the goal is to detect the types (labels) of all unknown relationships in the network. More precisely,

Problem 1. **Social relationship mining.** Given a partially labeled network $G = (V, E^L, E^U, R^L, \mathbf{W})$, the objective is to learn a predictive function

$$f : G = (V, E^L, E^U, R^L, \mathbf{W}) \rightarrow R$$

Our formulation of inferring social relationships is very different from existing works on relation mining [3]. They focus on detecting the relationships from the content information, while we focus on mining relationship semantics in social networks. Both Diehl et al.[4] and Wang et al.[26] investigate the problem of relationship identification. However, they focus on the problem in specific domains (Email network or Publication network).

3 Partially-Labeled Pairwise Factor Graph Model (PLP-FGM)

3.1 Basic Idea

In general, there are two ways to model the problem. The first way is to model each user as a node and for each node we try to estimate probability distributions of different relationships from the user to her neighborhood nodes in the social network. The graphical model consists of N variable nodes. Each node contains a $d \times |\mathcal{Y}|$ matrix to represent the probability distributions of different relationships between the user and her neighbors, where d is the number of neighbors of the node. This model is intuitive, but it suffers from some limitations. For example, it is difficult to model the correlations between two relationships, and

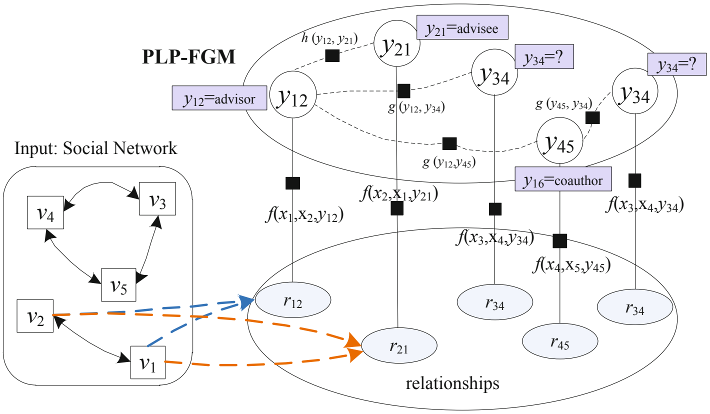


Fig. 2. Graphical representation of the PLP-FGM model

its computational complexity is high. An alternative way is to model each relationship as a node in the graphical model and the relationship mining task becomes how to predict the semantic label for each relationship node in the model. This model contains M nodes ($2M$ when the input social network is undirected). More importantly, this model is able to incorporate different correlations between relationships.

For inferring the type of social relationships, we have three basic intuitions. First, the user-specific or link-specific attributes will contain implicit information about the relationships. For example, two users who made a number of calls in working hours might be colleagues; while two users who frequently contact with each other in the evening are more likely to be family members or intimate friends. Second, relationships of different users may have a correlation. For example, in the mobile network, if user v_i makes a call to user v_j immediately after calling user v_k, then user v_i may have a similar relationship (family member or colleague) with user v_j and user v_k. Third, we need also consider some global constraints such as common knowledge or user-specific constraints.

3.2 Partially-Labeled Pairwise Factor Graph Model (PLP-FGM)

Based on the above intuitions, we propose a partially-labeled pairwise factor graph model (PLP-FGM). Figure 2 shows the graphical representation of the PLP-FGM. Each relationship (v_{i_1}, v_{i_2}) or $e_{i_1 i_2}$ in partially labeled network G is mapped to a *relationship node* r_i in PLP-FGM. We denote the set of relationship nodes as $Y = \{y_1, y_2, \ldots, y_M\}$. The relationships in G are partially labeled, thus all nodes in PLP-FGM can be divided into two subsets Y^L and Y^U, corresponding to the labeled and unlabeled relationships respectively. For each relationship node $y_i = (v_{i_1}, v_{i_2}, r_{i_1 i_2})$, we combine the attributes $\{\mathbf{w}_{i_1}, \mathbf{w}_{i_2}\}$ into a *relationship attribute vector* $\mathbf{x}_i$.

Now we explain the PLP-FGM in detail. The relationships in the input are modeled by relationship nodes in PLP-FGM. Corresponding to the three intuitions, we define the following three factors.

- *Attribute factor:* $f(y_i, \mathbf{x}_i)$ represents the posterior probability of the relationship y_i given the attribute vector $\mathbf{x}_i$;
- *Correlation factor:* $g(y_i, G(y_i))$ denotes the correlation between the relationships, where $G(y_i)$ is the set of correlated relationships to y_i.
- *Constraint factor:* $h(y_i, H(y_i))$ reflects the constraints between relationships, where $H(y_i)$ is the set of relationships constrained on y_i.

Given a partially-labeled network $G = (V, E^L, E^U, R^L, \mathbf{W})$, we can define the joint distribution over Y as

$$p(Y|G) = \prod_i f(y_i, \mathbf{x}_i) g(y_i, G(y_i)) h(y_i, H(y_i)) \tag{1}$$

The three factors can be instantiated in different ways. In this paper, we use exponential-linear functions. In particular, we define the attribute factor as

$$f(y_i, \mathbf{x}_i) = \frac{1}{Z_\lambda} \exp\{\lambda^T \mathbf{\Phi}(y_i, \mathbf{x}_i)\} \tag{2}$$

where λ is a weighting vector and $\mathbf{\Phi}$ is a vector of feature functions. Similarly, we define the correlation factor and constraint factor as

$$g(y_i, G(y_i)) = \frac{1}{Z_\alpha} \exp\{\sum_{y_j \in G(y_i)} \alpha^T \mathbf{g}(y_i, y_j)\} \tag{3}$$

$$h(y_i, H(y_i)) = \frac{1}{Z_\beta} \exp\{\sum_{y_j \in H(y_i)} \beta^T \mathbf{h}(y_i, y_j)\} \tag{4}$$

where $\mathbf{g}$ and $\mathbf{h}$ can be defined as a vector of indicator functions.

Model Learning. Learning PLP-FGM is to estimate a parameter configuration $\theta = (\lambda, \alpha, \beta)$, so that the log-likelihood of observation information (labeled relationships) are maximized. For presentation simplicity, we concatenate all factor functions for a relationship node y_i as $\mathbf{s}(y_i) = (\mathbf{\Phi}(y_i, \mathbf{x}_i)^T, \sum_{y_j} \mathbf{g}(y_i, y_j)^T, \sum_{y_j} \mathbf{h}(y_i, y_j)^T)^T$. The joint probability defined in (Eq. 1) can be written as

$$p(Y|G) = \frac{1}{Z} \prod_i \exp\{\theta^T \mathbf{s}(y_i)\} = \frac{1}{Z} \exp\{\theta^T \sum_i \mathbf{s}(y_i)\} = \frac{1}{Z} \exp\{\theta^T \mathbf{S}\} \tag{5}$$

where $Z = Z_\lambda Z_\alpha Z_\beta$ is a normalization factor (also called partition function), $\mathbf{S}$ is the aggregation of factor functions over all relationship nodes, i.e., $\mathbf{S} = \sum_i \mathbf{s}(y_i)$.

One challenge for learning the PLP-FGM model is that the input data is partially-labeled. To calculate the partition function Z, one needs to sum up the likelihood of possible states for all nodes including unlabeled nodes. To deal with

```
Input: learning rate $\eta$
Output: learned parameters $\theta$
Initialize $\theta$;
repeat
    Calculate $\mathbb{E}_{p_\theta(Y|Y^L,G)}\mathbf{S}$ using LBP ;
    Calculate $\mathbb{E}_{p_\theta(Y|G)}\mathbf{S}$ using LBP ;
    Calculate the gradient of $\theta$ according to Eq. 7:
        $\nabla_\theta = \mathbb{E}_{p_\theta(Y|Y^L,G)}\mathbf{S} - \mathbb{E}_{p_\theta(Y|G)}\mathbf{S}$
    Update parameter $\theta$ with the learning rate $\eta$:
        $\theta_{\text{new}} = \theta_{\text{old}} - \eta \cdot \nabla_\theta$
until Convergence;
```

Algorithm 1. Learning PLP-FGM

this, we use the labeled data to infer the unknown labels. Here $Y|Y^L$ denotes a labeling configuration Y inferred from the known labels. Thus, we can define the following log-likelihood objective function $\mathcal{O}(\theta)$:

$$\begin{aligned}\mathcal{O}(\theta) &= \log p(Y^L|G) = \log \sum_{Y|Y^L} \frac{1}{Z} \exp\{\theta^T \mathbf{S}\} \\ &= \log \sum_{Y|Y^L} \exp\{\theta^T \mathbf{S}\} - \log Z \\ &= \log \sum_{Y|Y^L} \exp\{\theta^T \mathbf{S}\} - \log \sum_{Y} \exp\{\theta^T \mathbf{S}\} \qquad (6)\end{aligned}$$

To solve the objective function, we can consider a gradient decent method (or a Newton-Raphson method). Specifically, we first calculate the gradient for each parameter θ:

$$\begin{aligned}\frac{\partial \mathcal{O}(\theta)}{\partial \theta} &= \frac{\partial \left(\log \sum_{Y|Y^L} \exp \theta^T \mathbf{S} - \log \sum_Y \exp \theta^T \mathbf{S}\right)}{\partial \theta} \\ &= \frac{\sum_{Y|Y^L} \exp \theta^T \mathbf{S} \cdot \mathbf{S}}{\sum_{Y|Y^L} \exp \theta^T \mathbf{S}} - \frac{\sum_Y \exp \theta^T \mathbf{S} \cdot \mathbf{S}}{\sum_Y \exp \theta^T \mathbf{S}} \\ &= \mathbb{E}_{p_\theta(Y|Y^L,G)}\mathbf{S} - \mathbb{E}_{p_\theta(Y|G)}\mathbf{S} \qquad (7)\end{aligned}$$

Another challenge here is that the graphical structure in PLP-FGM can be arbitrary and may contain cycles, which makes it intractable to directly calculate the second expectation $\mathbb{E}_{p_\theta(Y|G)\mathbf{S}}$. A number of approximate algorithms have been proposed, such as Loopy Belief Propagation (LBP) [17] and Mean-field [28]. In this paper, we utilize Loopy Belief Propagation. Specifically, we approximate marginal probabilities $p(y_i|\theta)$ and $p(y_i, y_j|\theta)$ using LBP. With the marginal probabilities, the gradient can be obtained by summing over all relationship nodes. It is worth noting that we need to perform the LBP process twice in each iteration, one time for estimating the marginal probability $p(y|G)$ and the other for $p(y|Y^L, G)$. Finally with the gradient, we update each parameter with a learning rate η. The learning algorithm is summarized in Algorithm 1.

Inferring Unknown Social Ties. We now turn to describe how to infer the type of unknown social relationships. Based on learned parameters θ, we can predict the label of each relationship by finding a label configuration which maximizes the joint probability (Eq. 1), i.e.,

$$Y^* = \text{argmax}_{Y|Y^L} p(Y|G) \tag{8}$$

Again, we utilize the loopy belief propagation to compute the marginal probability of each relationship node $p(y_i|Y^L, G)$ and then predict the type of a relationship as the label with largest marginal probability. The marginal probability is then taken as the prediction confidence.

3.3 Distributed Learning

As real social networks may contain millions of users and relationships, it is important for the learning algorithm to scale up well with large networks. To address this, we develop a distributed learning method based on MPI (Message Passing Interface). The learning algorithm can be viewed as two steps: 1) compute the gradient for each parameter via loopy belief propagation; 2) optimize all parameters with the gradient descents. The most expensive part is the step of calculating the gradient. Therefore we develop a distributed algorithm to speed up the process.

We adopt a *master-slave* architecture, i.e., one master node is responsible for optimizing parameters, and the other slave nodes are responsible for calculating gradients. At the beginning of the algorithm, the graphical model of PLP-FGM is partitioned into P roughly equal parts, where P is the number of slave processors. This process is accomplished by graph segmentation software METIS[11]. The subgraphs are then distributed over slave nodes. Note that in our implementation, the edges (factors) between different subgraphs are eliminated, which results in an approximate, but very efficient solution. In each iteration, the master node sends the newest parameters θ to all slaves. Slave nodes then start to perform Loopy Belief Propagation on the corresponding subgraph to calculate the marginal probabilities, then further compute the parameter gradient and send it back to the master. Finally, the master node collects and sums up all gradients obtained from different subgraphs, and updates parameters by the gradient descent method. The data transferred between the master and slave nodes are summarized in Table 1.

Table 1. Data transferred in distributed learning algorithm

Phase	From	To	Data Description
Initialization	Master	Slave i	i-th subgraph
Iteration Beginning	Master	Slave i	Current parameters θ
Iteration Ending	Slave i	Master	Gradient in i-th subgraph

Table 2. Statistics of three data sets

Data set	Users	Unlabeled Relationships	Labeled Relationships
Publication	1,036,990	1,984,164	6,096
Email	151	3,424	148
Mobile	107	5,122	314

4 Experimental Results

The proposed relationship mining approach is general and can be applied to many different scenarios. In this section, we present experiments on three different genres of data sets to evaluate the effectiveness and efficiency of our proposed approach. All data sets and codes are publicly available[1]

4.1 Experiment Setup

Data sets. We perform our experiments on three different data sets: Publication, Email, and Mobile. Statistics of the data sets are shown in Table 2.

- **Publication.** In the publication data set, we try to infer the advisor-advisee relationship from the coauthor network. The data set is provided by [26]. Specifically, we have collected 1,632,442 publications from Arnetminer [24] (from 1936 to 2010) with 1,036,990 authors involved. The ground truth is obtained in three ways: 1) manually crawled from researcher's homepage; 2) extracted from Mathematics Genealogy project[2]; 3) extracted from AI Genealogy project[3]. In total, we have collected 2,164 advisor-advisee pairs as positive cases, and another 3,932 pairs of colleagues as negative cases. The mining results for advisor-advisee relationships are also available in the online system Arnetminer.org.
- **Email.** In the email data set, we aim to infer the manager-subordinate relationship from the email communication network. The data set consists of 136,329 emails between 151 Enron employees. The ground truth of manager-subordinate relationships is provided by [4].

[1] http://arnetminer.org/socialtie/

[2] `http://www.genealogy.math.ndsu.nodak.edu`

[3] `http://aigp.eecs.umich.edu`

- **Mobile.** In the mobile data set, we try to infer the friendship in mobile calling network. The data set is from Eagle et al. in [6]. It consists of call logs, bluetooth scanning logs and location logs collected by a software installed in mobile phones of 107 users during a ten-month period. In the data set, users provide labels for their friendships. In total, 314 pairs of users are labeled as friends.

Factor definition. In the Publication data set, relationships are established between authors v_i and v_j if they coauthored at least one paper. For each pair of coauthors (v_i, v_j), our objective is to identify whether v_i is the advisor of author v_j. In this data set, we consider two types of correlations: 1) *co-advisee*. The assumption is based on the fact that one could have only a limited number of advisors in her/his research career. Based on this, we define a correlation factor h_1 between nodes r_{ij} and r_{kj}. 2) *co-advisor*. Another observation is that if v_i is the advisor of v_j (i.e., $r_{ij} = 1$), then v_i is very possible to be the advisor of some other student v_k who is similar to v_j. We define another factor function h_2 between nodes r_{ij} and r_{ik}.

In the Email data set, we try to discover the "manager-subordinate" relationship. A relationship (v_i, v_j) is established when two employees have at least one email communication. There are in total 3,572 relationships among which 148 are labeled as manager-subordinate relationships. We try to identify the relationship types from the email traffic network. For example, if most of an employee's emails were sent to the same one, then the recipient is very likely to be her manager. A correlation named *co-recipient* is defined, that is, if a user v_i sent more than ϑ emails of which recipients including both v_j and v_k (ϑ is a threshold and is set as 10 in our experiment), then, the relationship r_{ij} and r_{ik} are very likely to be the same. Therefore, a correlation factor is added between the two relationships. Two constraints named *co-manager* and *co-subordinate* are also introduced in an analogous way as that for the publication data.

In the Mobile data set, we try to identify whether two users have a friendship if there were at least one voice call or one text message sent from one to the other. Two kinds of correlations are considered: 1) *co-location*: if more than three users arrived in the same location roughly the same time, we establish correlations between all the relationships in this groups. 2) *related-call*. When v_i makes a call to both v_k and v_j from the same location, or makes a call to v_k immediately after the call with v_j, we add a related-call correlation factor between r_{ij} and r_{ik}.

In addition, we also consider some other features in the three data sets. A detailed description of the factor definition for each data set is given in Table 5 in Appendix.

Comparison methods. We compare our approach with the following methods for inferring relationship types:

SVM: It uses the relationship attribute vector $\mathbf{x}_i$ to train a classification model, and predict the relationships by employing the classification model. We use the SVM-light package to implement SVM.

Table 3. Performance of relationship mining with different methods on three data sets: Publication, Email and Mobile (%)

Data set	Method	Accuracy	Precision	Recall	F1-score
Publication	SVM	76.6	72.5	54.9	62.1
	TPFG	81.2	82.8	**89.4**	86.0
	PLP-FGM-S	84.1	77.1	78.4	77.7
	PLP-FGM	**92.7**	**91.4**	87.7	**89.5**
Email	SVM	82.6	79.1	**88.6**	83.6
	PLP-FGM-S	85.6	85.8	85.6	85.7
	PLP-FGM	**88.0**	**88.6**	87.2	**87.9**
Mobile	SVM	80.0	**92.7**	64.9	76.4
	PLP-FGM-S	80.9	88.1	71.3	78.8
	PLP-FGM	**83.1**	89.4	**75.2**	**81.6**

TPFG: It is an unsupervised method proposed in [26] for mining advisor-advisee relationships in publication network. This method is domain-specific and thus we only compare with it on the Publication data set.

PLP-FGM-S: The proposed PLP-FGM is based on the partially-labeled network. Another alternative strategy is to train the model (parameters) with the labeled nodes only. We use this method to evaluate the necessity of the partial learning.

Evaluation measures. To quantitatively evaluate the proposed method, we consider two aspects: performance and scalability. For the relationship mining performance, we consider two-fold cross-validation(i.e., half training and half testing) and evaluate the approaches in terms of accuracy, precision, recall, and F1-score. For scalability, we examine the execution time of the model learning.

All the codes are implemented in C++, and all experiments are conducted on a server running Windows Server 2008 with Intel Xeon CPU E7520 1.87GHz (16 cores) and 128 GB memory. The distributed learning algorithm is implemented on MPI (Message Passing Interface).

4.2 Accuracy Performance

Table 3 lists the accuracy performance of inferring the type of social relationships by the different methods.

Performance comparison. Our method consistently outperforms other comparative methods on all the three data sets. In the Publication data set, PLP-FGM achieves a +27% (in terms of F1-score) improvement compared with SVM, and outperforms TPFG by 3.5% (F1-score) and 11.5% in terms of accuracy. We observe that TPFG achieves the best recall among all the four methods. This is because that TPFG tends to predict more positive cases (i.e., inferring more advisor-advisee relationships in the coauthor network), thus would hurt the precision. As a result, TPFG underperforms our method 8.6% in terms of precision.

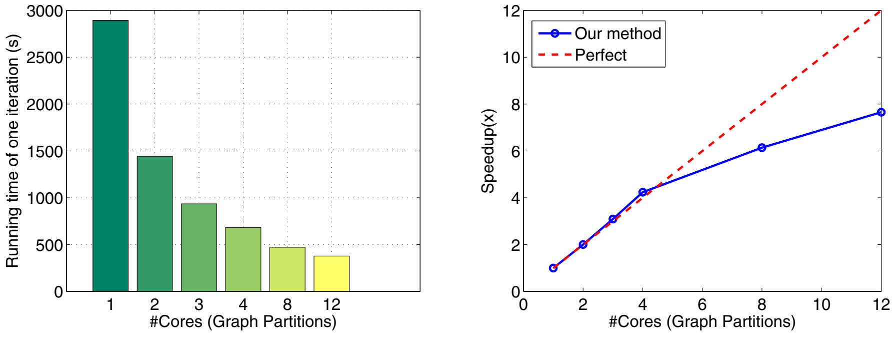
Table 4. Factor contribution analysis on three data sets. (%)

Data set	Factors used	Accuracy	Precision	Recall	F1-score
Publication	Attributes	77.1	71.1	59.8	64.9
	+ Co-advisor	83.5	80.9	69.8	75.0 (+10.1%)
	+ Co-advisee	83.1	79.7	70.2	74.7 (+9.8%)
	All	92.7	91.4	87.7	89.5(+24.6%)
Email	Attributes	80.1	79.5	81.2	80.3
	+ Co-recipient	80.8	81.5	79.7	80.6 (+0.3%)
	+ Co-manager	83.1	82.8	83.5	83.2 (+2.9%)
	+ Co-subordinate	85.0	84.4	85.7	85.0 (+4.7%)
	All	88.0	88.6	87.2	87.9 (+7.6%)
Mobile	Attributes	81.8	88.6	73.3	80.2
	+ Co-location	82.2	89.2	73.3	80.4 (+0.2%)
	+ Related-call	81.8	88.6	73.3	80.2 (+0.0%)
	All	83.1	89.4	75.2	81.6 (+1.4%)

In Email and Mobile data set, PLP-FGM outperforms SVM by +4% and +5% respectively.

Unlabeled data indeed helps. From the result, it clearly showed that by utilizing the unlabeled data, our model indeed obtains a significant improvement. Without using the unlabeled data, our model (PLP-FGM-S) results in a large performance reduction (-11.8% in terms of F1-score) on the publication data set. On the other two data sets, we also observe a clear performance reduction.

Factor contribution analysis. We perform an analysis to evaluate the contribution of different factors defined in our model. We first remove all the correlation/constraint factors and only keep the attribute factor, and then add each of the factors into the model and evaluate the performance improvement by each factor. Table 4 shows the result of factor analysis. We see that almost all the factors are useful for inferring the social relationships, but the contribution is very different. For example, for inferring the manager-subordinate relationship, the co-subordinate factor is the most useful factor which achieves a 4.7% improvement by F1-score, and the co-manager factor achieves a 2.9% improvement; while the co-recipient factor only results in a 0.3% improvement. However, by combining all the factors together, we can further obtain a 2.9% improvement. An extreme phenomenon appears on the Mobile data set. With each of the two factors (co-location and related-call), we cannot obtain a clear improvement (0.2% and 0.0% by F1). However, when combining the two factors and the attribute factor together, we can achieve a 1.4% improvement. This is because our model not only considers different factors, but also leverages the correlation between them.



(a) Running time vs. #cores (b) Speedup vs. #cores

Fig. 3. Scalability performance

4.3 Scalability Performance

We now evaluate the scalability performance of our distributed learning algorithm on the Publication data set. Figure 3 shows the running time and speedup of the distributed algorithm with different number of computer nodes (2,3,4,8,12 cores) used. The speedup curve is close to the perfect line at the beginning. Although the speedup inevitably decreases when the number of cores increases, it can achieve $\sim 8\times$ speedup with 12 cores. It is noticeable that the speedup curve is beyond the perfect line when using 4 cores, it is not strange since our distributed strategy is approximated. In our distributed implementation, graphs are partitioned into subgraphs, and the factors across different parts are discarded. Thus, the graph processed in distributed version contains less edges, making the computational cost less than the amount in the original algorithm. The effect of subgraph partition is illustrated in Figure 4. By using good graph partition algorithm such as METIS, the performance only decreases slightly (1.4% in accuracy and 1.6% in F1-score). A theoretical study of the approximate ratio for

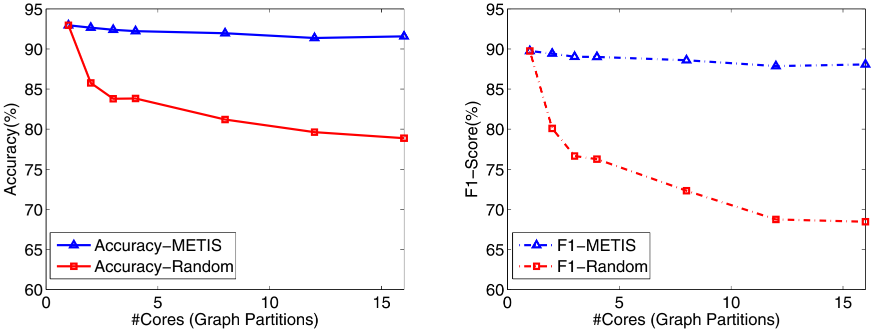


Fig. 4. Approximation of graph partition

the distributed learning algorithm would be an interesting issue and is also one of our ongoing work.

5 Related Work

Relationship mining is an important problem in social network analysis. One research branch is to predict and recommend unknown links in social networks. Liben-Nowell et al.[16] study the unsupervised methods for link prediction. Xiang et al. [27] develop a latent variable model to estimate relationship strength from interaction activity and user similarity. Backstrom et al. [2] propose a supervised random walk algorithm to estimate the strength of social links. Leskovec et al. [15] employ a logistic regression model to predict positive and negative links in online social networks, where the positive links indicates the relationships such as friendship, while negative indicating opposition. However, these works consider only the black-white social networks, and do not consider the types of the relationships. There are also several works on mining the relationship semantics. Diehl et al. [4] try to identify the manager-subordinate relationships by learning a ranking function. Wang et al. [26] propose an unsupervised probabilistic model for mining the advisor-advisee relationships from the publication network. Eagle et al. [6] present several patterns discovered in mobile phone data, and try to use these pattern to infer the friendship network. However, these algorithms mainly focus on a specific domain, while our model is general and can be applied to different domains. Moreover, these methods do not explicitly consider the correlation information between different relationships.

Another related research topic is relational learning[3,8]. However, the problem presented in this paper is very different. Relational learning focuses on the classification problems when objects or entities are presented in relations, while this paper explores the relationship types in social network. A number of supervised methods for link prediction in relational data have also been developed [25,19].

6 Conclusion

In this paper, we study the problem of inferring the type of social ties in large networks. We formally define the problem in a semi-supervised framework, and propose a partially-labeled pairwise factor graph model (PLP-FGM) to learn to infer the relationship semantics. In PLP-FGM, relationships in social network are modeled as nodes, the attributes, correlations and global constraints are modeled as factors. An efficient algorithm is proposed to learn model parameters and to predict unknown relationships. Experimental results on three different types of data sets validate the effectiveness of the proposed model. To further scale up to large networks, a distributed learning algorithm is developed. Experiments demonstrate good parallel efficiency of the distributed learning algorithm.

Detecting the relationship semantics makes online social networks colorful and closer to our real physical networks. It represents a new research direction in social network analysis. As future work, it is interesting to study how to further improve the mining performance by involving users into the learning process (e.g., via active learning). In addition, it would be also interesting to investigate how the inferred relationship semantic information can help other applications such as community detection, influence analysis, and link recommendation.

References

1. Albert, R., Barabasi, A.L.: Statistical mechanics of complex networks. Reviews of Modern Physics 74(1) (2002)
2. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: WSDM, pp. 635–644 (2011)
3. Califf, M.E., Mooney, R.J.: Relational learning of pattern-match rules for information extraction. In: AAAI/IAAI, pp. 328–334 (1999)
4. Diehl, C.P., Namata, G., Getoor, L.: Relationship identification for social network discovery. In: AAAI, pp. 546–552. AAAI Press, Menlo Park (2007)
5. Domingos, P., Richardson, M.: Mining the network value of customers. In: KDD, pp. 57–66 (2001)
6. Eagle, N., Pentland, A.S., Lazer, D.: Mobile phone data for inferring social network structure. In: Social Computing, Behavioral Modeling, and Prediction, pp. 79–88 (2008)
7. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. In: SIGCOMM, pp. 251–262 (1999)
8. Getoor, L., Taskar, B.: Introduction to statistical relational learning. The MIT Press, Cambridge (2007)
9. Granovetter, M.: The strength of weak ties. American Journal of Sociology 78(6), 1360–1380 (1973)
10. Grob, R., Kuhn, M., Wattenhofer, R., Wirz, M.: Cluestr: mobile social networking for enhanced group communication. In: GROUP, pp. 81–90 (2009)
11. Karypis, G., Kumar, V.: MeTis: Unstrctured Graph Partitioning and Sparse Matrix Ordering System, version 4.0 (September 1998)
12. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
13. Kleinberg, J.: Temporal dynamics of on-line information streams. In: Data Stream Managemnt: Processing High-Speed Data. Springer, Heidelberg (2005)
14. Krackhardt, D.: The Strength of Strong Ties: The Importance of Philos in Organizations, pp. 216–239. Harvard Business School Press, Boston
15. Leskovec, J., Huttenlocher, D.P., Kleinberg, J.M.: Predicting positive and negative links in online social networks. In: WWW, pp. 641–650 (2010)
16. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. JASIST 58(7), 1019–1031 (2007)
17. Murphy, K., Weiss, Y., Jordan, M.: Loopy belief propagation for approximate inference: An empirical study. In: UAI, vol. 9, pp. 467–475 (1999)
18. Newman, M.E.J.: The structure and function of complex networks. SIAM Reviews 45 (2003)
19. Popescul, A., Ungar, L.: Statistical relational learning for link prediction. In: IJCAI 2003 Workshop on Learning Statistical Models from Relational Data, vol. 149, p. 172 (2003)

20. Roth, M., Ben-David, A., Deutscher, D., Flysher, G., Horn, I., Leichtberg, A., Leiser, N., Matias, Y., Merom, R.: Suggesting friends using the implicit social graph. In: KDD, pp. 233–242 (2010)
21. Strogatz, S.H.: Exploring complex networks. Nature 410, 268–276 (2003)
22. Tan, C., Tang, J., Sun, J., Lin, Q., Wang, F.: Social action tracking via noise tolerant time-varying factor graphs. In: KDD, pp. 1049–1058 (2010)
23. Tang, J., Sun, J., Wang, C., Yang, Z.: Social influence analysis in large-scale networks. In: KDD, pp. 807–816 (2009)
24. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: Extraction and mining of academic social networks. In: KDD 2008, pp. 990–998 (2008)
25. Taskar, B., Wong, M.F., Abbeel, P., Koller, D.: Link prediction in relational data. In: NIPS. MIT Press, Cambridge (2003)
26. Wang, C., Han, J., Jia, Y., Tang, J., Zhang, D., Yu, Y., Guo, J.: Mining advisor-advisee relationships from research publication networks. In: KDD, pp. 203–212 (2010)
27. Xiang, R., Neville, J., Rogati, M.: Modeling relationship strength in online social networks. In: WWW, pp. 981–990 (2010)
28. Xing, E.P., Jordan, M.I., Russell, S.: A generalized mean field algorithm for variational inference in exponential families. In: UAI 2003, pp. 583–591 (2003)

Appendix: Feature Definition

In this section, we introduce how we define the attribute factor functions. In the Publication data set, we define five categories of attribute factors: Paper count, Paper ratio, Coauthor ratio, Conference coverage, First-paper-year-diff. The definitions of the attributes are summarized in Table 5. In the Email data set, traffic-based features are extracted. For a relationship, we compute the number of emails for different communication types. In the Mobile data set, the attributes we extracted are #voice calls, #messages, Night-call ratio, Call duration, #proximity and In-role proximity ratio.

Table 5. Attributes used in the experiments. In the Publication data set, we use P_i and P_j to denote the set of papers published by author v_i and v_j respectively. For a given relationship (v_i, v_j), five categories of attributes are extracted. In the Email data set, for relationship (v_i, v_j), number of emails for different communication types are computed. In the Mobile data set, the attributes are from the voice call/message/proximity logs.

Data set	Factor	Description	
Publication	Paper count	$\|P_i\|$, $\|P_j\|$	
	Paper ratio	$\|P_i\|/\|P_j\|$	
	Coauthor ratio	$\|P_i \cap P_j\|/\|P_i\|$, $\|P_i \cap P_j\|/\|P_j\|$	
	Conference coverage	The proportion of the conferences which both v_i and v_j attended among conferences v_j attended.	
	First-paper-year-diff	The difference in year of the earliest publication of v_i and v_j.	
Email	Traffics	Sender	Recipients Include
		v_i	v_j
		v_j	v_i
		v_i	v_k and not v_j
		v_j	v_k and not v_i
		v_k	v_i and not v_j
		v_k	v_j and not v_i
		v_k	v_i and v_j
Mobile	#voice calls	The total number of voice call logs between two users.	
	#messages	Number of messages between two users.	
	Night-call ratio	The proportion of calls at night (8pm to 8am).	
	Call duration	The total duration time of calls between two users.	
	#proximity	The total number of proximity logs between two users.	
	In-role proximity ratio	The proportion of proximity logs in "working place" and in working hours (8am to 8pm).	

Comparing Apples and Oranges
Measuring Differences between Data Mining Results

Nikolaj Tatti and Jilles Vreeken

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

Abstract. Deciding whether the results of two different mining algorithms provide significantly different information is an important open problem in exploratory data mining. Whether the goal is to select the most informative result for analysis, or decide which mining approach will likely provide the most novel insight, it is essential that we can tell how different the information is that two results provide.

In this paper we take a first step towards comparing exploratory results on binary data. We propose to meaningfully convert results into sets of noisy tiles, and compare between these sets by Maximum Entropy modelling and Kullback-Leibler divergence. The measure we construct this way is flexible, and allows us to naturally include background knowledge, such that differences in results can be measured from the perspective of what a user already knows. Furthermore, adding to its interpretability, it coincides with Jaccard dissimilarity when we only consider exact tiles.

Our approach provides a means to study and tell differences between results of different data mining methods. As an application, we show that it can also be used to identify which parts of results best redescribe other results. Experimental evaluation shows our measure gives meaningful results, correctly identifies methods that are similar in nature, and automatically provides sound redescriptions of results.

1 Introduction

Deciding whether the results of different mining algorithms provide significantly different information is an important, yet understudied, open problem in exploratory data mining. Whether we want to select the most promising result for analysis by an expert, or decide which mining approach we should apply next in order to most likely gain most novel insight, we need to be able to tell how different the information is that different results, by possibly different methods, provide. However, while the comparison of results is a well-studied topic in statistics, it has received much less attention in the knowledge discovery community.

Clearly, any dataset only contains a limited amount of knowledge—which is the most that we can hope to discover from it. To extract this information, we have an ever growing number of data mining algorithms at our disposal. However, most data mining results are complex, and their analysis and validation often

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 398–413, 2011.

takes considerable effort and cost. So, simply applying ‘all’ methods and letting an expert analyse ‘all’ results is not a feasible approach to extract ‘all’ knowledge. Moreover, many of these results will be redundant, i.e. convey roughly the same information, and hence only require effort while not providing extra insight.

Instead, we would ideally just select that result for analysis which will provide us the most new knowledge. In order to be able to do this, two basic requirements have to be met. First of all, we need to be able to measure how different two results are from an information-providing perspective; if they essentially provide the same information, we could just select one for processing. Second, we should be able to include our background knowledge, such that we can gauge the amount of information a result gives us compared to what we already know.

Although an important practical problem, it has been surprisingly understudied in data mining. The main focus in exploratory data mining research has mostly been on developing techniques to discover structure, and, not so much on how to compare between results of different methods. As a result there currently exist no general methods or theory to this end in the data mining literature.

For tasks where a formal objective is available, we can straightforwardly use it to compare fairly between the results of different methods. In classification, for instance, we can use accuracy. For exploratory data mining, however, there is no formal common goal: any result that provides novel insight is potentially useful. The core of the problem is thus that comparing between methods is like comparing *apples* to *oranges*: a clustering is a different result than a set of itemsets, which are, in turn, different from a classifier, set of subgroups, etc. So, in order to make a sensible comparison, we need to find a common language.

In this regard, the comparison of complex objects, e.g. of datasets [23,25], is related. Our setting, however, is more general, as now we do not want to compare between one type of complex object, but want to consider a very rich class of objects—potentially consisting of any data mining result. Arguably, some of the most general complex objects to compare between are probability distributions. Statistics and Information Theory provide us tools for measuring differences between distributions, such as Kullback-Leibler divergence [2]. Mining results, however, rarely are probability distributions, and if they are, not necessarily for the same random variable; making these tools unsuited for direct application.

A simple yet important observation we make is that any mining result essentially identifies some properties of the dataset at hand. In an abstract way, we could identify all datasets for which these properties hold. This is an important notion, as it provides us a way to compare between results of different methods: if two results provide the same information, they identify the same subspace of possible datasets, and the more different the information two results give, the smaller the overlap between the sets of possible datasets will be. More generally put: every data mining result implicitly defines a probability distribution over datasets. And hence, if we can model these distributions, we can use standard tools from Statistics to compare between data mining results fair and square.

In this paper, we propose to translate data mining results into probability distributions over datasets using the Maximum Entropy principle [3]. It allows us to uniquely identify the model that makes optimal use of the provided information, but is fully unbiased otherwise. By subsequently measuring the Kullback-Leibler divergence between these distributions, we can tell how different two results are from an information perspective. Besides data mining results, we can also incorporate background knowledge into our model, and so use it to score results from specific points of view [5].

Finding these probability distributions, and conditioning them using data mining results, however, is far from trivial. Here, we therefore give a proof of concept of our approach for binary data, for which the basics of maximum entropy modelling are available. We show that many exploratory data mining results on binary data can easily be translated into sets of noisy tiles: combinations of rows and columns, for which we know the density of 1s. We show we can efficiently acquire the maximum entropy distribution given sets of such tiles, and that by KL divergence we can so compare between results.

More specifically, in our experiments we compare between the results of ten different exploratory data mining methods, including (bi-)clusters, subspace clusters, sets of tiles, and sets of frequent itemsets. Moreover, we give a theoretic framework to mine for redescriptions. That is, given a (sub)set of noisy tiles from one result, we can identify the set of tiles from another result that best approximates the same information. Experiments show our measure works well in practice: dissimilarity converges to 0 when models approximate each other, methodologically close methods are correctly grouped together, and sensible redescriptions for tile-sets are obtained. In other words, we give an approach by which we can meaningfully *mix* apples and oranges, and compare them fairly.

The roadmap of this paper is as follows. Next, in Section 2 we give the notation and preliminaries we use throughout the paper. Section 3 details how we can build a global model from a set of tiles, which we use in Section 4 to define a measure to compare such sets. In Section 5 we subsequently use this measure for redescribing sets of tiles. We discuss related work in Section 6, and we evaluate our measure empirically in Section 7. We round up with discussion and conclusions in Sections 8 and 9. Due to lack of space, we give the proofs, such as for NP-completeness, in the Appendix [24].

2 Preliminaries

In this section, we define the preliminaries we will use in subsequent sections.

A *binary dataset* D is a binary matrix of size $N \times M$ consisting of N rows, binary vectors of size M. We denote $(i, j)^{\text{th}}$ entry of D by $D(i, j)$. We denote the space of all binary datasets of size $N \times M$ by $\mathcal{D}$.

We approach the comparison of different data mining results by first translating these into sets of tiles. A *tile* $T = (t(T), a(T))$ is a tuple consisting of two lists. The first list, $t(T)$, is a set of integers between 1 and N representing the transactions of T. The second list, $a(T)$, is a set of integers between 1 and M representing the attributes. We define $s(T)$ to be the cartesian product of $t(T)$ and $a(T)$, $s(T) = \{(i, j) \mid i \in t(T), j \in a(T)\}$. Given a tile set $\mathcal{T}$ we also define $s(\mathcal{T}) = \bigcup_{T \in \mathcal{T}} s(T)$.

Given a tile T and a dataset D we define a frequency $fr(T;D)$ to be the proportion of ones in D corresponding to the entries identified by T,

$$fr(T;D) = \frac{1}{|s(T)|} \sum_{i \in t(T)} \sum_{j \in a(T)} D(i,j) \quad .$$

There are numerous techniques for mining tile sets but we can also naturally describe a large number of statistics and mining results using tile sets:

- *density*: the frequency of a tile containing the whole data is equal to the density of the data.
- *margins*: the frequency of a column i can be expressed with a single (noisy) tile containing the column i and all transactions. Analogously, we can express the margins for each row.
- *itemsets*: any itemset can be converted into a tile by taking the supporting transactions. Thus, an itemset collection can be converted into a tile set.
- *bi/subspace-clustering*: subspace and bi-clusters are sets of transactions and columns. Hence, we can naturally represent these results by equivalent tiles.
- *clustering*: Given a clustering, either over transactions or items, we can construct a tile set in two different ways. The first way is to represent each cluster by a single tile, representing the density of a tile. The other way is to compute (column) margins for each cluster and represent these margins by tiles. This is particularly natural for k-means, since a centroid then corresponds to the column margins of the corresponding transactions.

Let p be a distribution defined over $\mathcal{D}$, the space of all datasets of size $N \times M$. We define the frequency of a tile to be the average frequency with respect to p,

$$fr(T;p) = \sum_{D \in \mathcal{D}} p(D) fr(T;D) \quad .$$

We can also express the frequency directly by this distribution.

Lemma 1. *Given a distribution p and a tile T, the frequency is equal to*

$$fr(T;p) = \sum_{(i,j) \in s(T)} p((i,j) = 1),$$

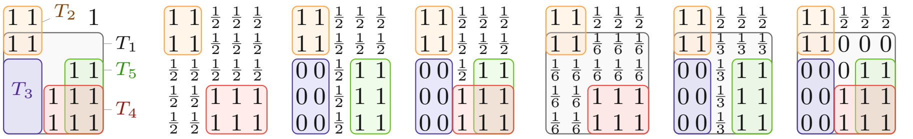
where $p((i,j) = 1)$ is the probability of a dataset having 1 as (i,j)th entry.

We say a tile is *exact* if its frequency is 0 or 1, and otherwise say it is *noisy.*

Corollary 1. *For an exact tile T, $p((i,j) = 1) = fr(T;p)$, where $(i,j) \in s(T)$.*

Example 1. Consider a dataset D given in Figure 1(a). We consider five different tiles, $T_1 = (2,\ldots,5) \times (1,\ldots,5)$, $T_2 = (1,2) \times (1,2)$, $T_3 = (3,4,5) \times (1,2)$, $T_4 = (4,5) \times (3,4,5)$, and $T_5 = (3,4,5) \times (4,5)$. The frequencies are $fr(T_1;D) = 10/20 = 1/2$, $fr(T_2;D) = fr(T_4;D) = fr(T_5;D) = 1$, and $fr(T_3;D) = 0$. By definition, $T_2,\ldots,T_5$ are exact, while T_1 is not.

Given two distributions, say p and q, we resp. define entropy and Kullback-Leibler divergence as



(a) D (b) $\mathcal{T}$ (c) $\mathcal{U}$ (d) $\mathcal{T} \cup \mathcal{U}$ (e) $\mathcal{T} \cup \mathcal{B}$ (f) $\mathcal{U} \cup \mathcal{B}$ (g) $\mathcal{T}\cup\mathcal{U}\cup\mathcal{B}$

Fig. 1. Toy example of a dataset and several maximum entropy models

$$H(p) = -\sum_{D \in \mathcal{D}} p(D) \log p(D) \quad \text{and} \quad KL(p \,\|\, q) = \sum_{D \in \mathcal{D}} p(D) \log \frac{p(D)}{q(D)}.$$

3 Building Global Models from Tiles

To meet our goal, we have to construct a statistically sound technique for comparing two sets of tiles. In this section we construct a global model for datasets using the given tiles. We will use these models for comparing the tile sets.

Consider that we are given a tile set $\mathcal{T}$, and for each tile $T \in \mathcal{T}$ we are also given a frequency α_T. Typically, the frequencies are obtained from the data at hand, $\alpha_T = fr(T; D_{in})$, but this is not a necessary condition. The tiles convey local information about the data D_{in} and our goal is to infer a distribution p over $\mathcal{D}$, that is, how probable data set $D \in \mathcal{D}$ is given a tile set $\mathcal{T}$. If the information at hand defines the data set uniquely, then $p(D) = 1$ if and only if $D = D_{in}$.

To derive the model, we use a well-founded notion from information theory, the Maximum Entropy principle [2]. Roughly speaking, by Maximum Entropy, we incorporate the given information into a distribution, yet further making it as evenly spread as possible. To define the distribution, we first define the space of distribution candidates. That is, the space of those distributions that produce the same frequencies for the given tiles, $\mathcal{P} = \{p \mid fr(T; p) = \alpha_T, \text{ for all } T \in \mathcal{T}\}$. In other words, $\mathcal{P}$ contains all distributions that explain the frequencies α_T. From this set, we select one distribution, which we denote by $p^*_{\mathcal{T}}$, such that $p^*_{\mathcal{T}}$ maximises the entropy, $H(p^*_{\mathcal{T}}) \geq H(p)$ for any $p \in \mathcal{P}$.

We will abuse notation and write $H(\mathcal{T})$ where we mean $H(p^*_{\mathcal{T}})$. Similarly we write $KL(\mathcal{T} \,\|\, \mathcal{U})$ to mean $KL(p^*_{\mathcal{T}} \,\|\, p^*_{\mathcal{U}})$, where $\mathcal{U}$ is another tile set.

A classic theorem states that p^* can be written as an exponential form.

Theorem 1 (Theorem 3.1 in [3]). *Given a tile set $\mathcal{T}$, a distribution p^* is the maximum entropy distribution if and only if it can be written as*

$$p^*(D) \propto \begin{cases} \exp\left(\sum_{T \in \mathcal{T}} \lambda_T fr(T; D)\right) & D \notin \mathcal{Z} \\ 0 & D \in \mathcal{Z}, \end{cases}$$

where λ_T is a certain weight for $fr(T; D)$ and $\mathcal{Z}$ is a collections of datasets such that $p(D) = 0$ for each $p \in \mathcal{P}$.

Algorithm 1. Iterative Scaling for solving the MaxEnt distribution

```
input  : tile set $\mathcal{T}$, target frequencies $\{\alpha_T\}$
output : Maximum entropy distribution $p$
$p \leftarrow$ a matrix of size $N \times M$ with values $1/2$;
foreach $T \in \mathcal{T}$, $\alpha_T = 0, 1$ do  $p(i,j) \leftarrow \alpha_T$ for all $(i,j) \in s(T)$;
while not converged do
    foreach $T \in \mathcal{T}$, $0 < \alpha_T < 1$ do
        $f \leftarrow fr(T; p)$;
        $x \leftarrow (\alpha_T(1 - f))/(f(1 - \alpha_T))$;
        $p(i,j) \leftarrow p(i,j)x/(1 - p(i,j)(1 - x))$ for all $(i,j) \in s(T)$;
```

The next theorem allows to factorize the distribution p^* into a product of Bernoulli random variables, each variable representing a single entry in the dataset. Such a representation gives us a practical way for inferring the model.

Theorem 2. *Let $\mathcal{T}$ be a tile set. Write $\mathcal{T}(i,j) = \{T \in \mathcal{T} \mid (i,j) \in s(T)\}$ to be the subset of $\mathcal{T}$ containing the tiles that cover an entry (i,j). Then, the maximum entropy distribution can be factorized as $p^*(D) = \prod_{i,j} p^*((i,j) = D(i,j))$, where*

$$p^*((i,j) = 1) = \frac{\exp\left(\sum_{T \in \mathcal{T}(i,j)} \lambda_T\right)}{\exp\left(\sum_{T \in \mathcal{T}(i,j)} \lambda_T\right) + 1} \quad \textit{or} \quad p^*((i,j) = 1) = 0, 1 \quad .$$

Theorem 2 allows to represent p^* as Bernoulli variables. We should stress that this is a different model than assuming independence between items in a random transaction. Our next step is to discover the correct frequencies for these variables. Here we use a variant of a well-known Iterative Scaling algorithm [4]. The algorithm is given in Algorithm 1. Informally said, given a tile T the algorithm updates the probabilities such that the frequency of T is closer to α_T. This is performed for each tile. Updating a single tile might change the frequency of another tile. Hence, we need several passes. A single pass takes $O(NM)$ time. The original proof of correctness for iterative scaling assumes that p^* has no zero probabilities. This is often violated in our setup. Hence we provide a proof of correctness in the Appendix.

Theorem 3. *The iterative scaling algorithm given in Algorithm 1 correctly converges to the maximum entropy distribution.*

It turns out, that if the given tiles are exact, the maximum entropy distribution is simple. A Bernoulli variable corresponding to the (i,j) entry is always 1 (or 0), if the entry is covered by an exact tile, i.e. with frequency 1 (or 0). Otherwise, the variable is equal to a fair coin toss. This form will allow us to express distances between sets of exact tiles in the next section.

Theorem 4. *Let $\mathcal{T}$ be a collection of exact tiles and let α_T be the desired frequency of a tile $T \in \mathcal{T}$. Then*

$$p^*_{\mathcal{T}}((i,j)=1) = \begin{cases} \alpha_T & \textit{if there exists } T \in \mathcal{T} \textit{ such that } (i,j) \in s(T) \\ 1/2 & \textit{otherwise} \end{cases} .$$

Example 2. Let us continue Example 1. Consider the following three tile sets $\mathcal{T} = \{T_2, T_4\}$, $\mathcal{U} = \{T_2, T_3, T_5\}$, and $\mathcal{B} = \{T_1\}$. The corresponding maximum entropy models are given in Figure 1, such that each entry represents the probability $p^*((i,j)=1)$. As the sets $\mathcal{T}$ and $\mathcal{U}$ contain only exact tiles, by Theorem 4 the entries for the models $p^*_{\mathcal{T}}$, $p^*_{\mathcal{U}}$, and $p^*_{\mathcal{T}\cup\mathcal{U}}$ are either 0, 1, or 1/2.

Consider $p^*_{\mathcal{T}\cup\mathcal{B}}$. Corollary 1 states that entries in $s(T_2)$ and $s(T_4)$ should be 1, since both tiles are exact. From T_1, we know that there are 10 ones, yet T_2 and T_4 account only for 8. Hence, there should be 2 ones in the 12 entries outside of $\mathcal{T}$, on average. We aim to be as fair as possible, hence we spread uniformly, giving us probabilities $2/12 = 1/6$. For $p^*_{\mathcal{U}\cup\mathcal{B}}$, there are 2 unaccounted 1s in 6 entries, giving us $2/6 = 1/3$. Finally, all 1s are accounted for in $p^*_{\mathcal{T}\cup\mathcal{U}\cup\mathcal{B}}$. Outside of $\mathcal{T} \cup \mathcal{U} \cup \mathcal{B}$ we have no information, hence these probabilities default to 1/2.

4 Comparing Sets of Tiles

Now that we have a technique for incorporating the information contained within a set of tiles into a model, we can use these models to compare sets of tiles.

We assume that we are given three tile sets $\mathcal{T}$, $\mathcal{U}$, and $\mathcal{B}$. Let tile set $\mathcal{B}$ contain the background information. Typically, this information would be simple, like column margins, row margins, or just the proportions of ones in the whole dataset. $\mathcal{B}$ can be also be empty, if we do not have or wish to use any background knowledge. Our goal is now to compute the distance between $\mathcal{T}$ and $\mathcal{U}$ given $\mathcal{B}$. We assume that the frequencies for the tile sets we are given are mutually consistent; which is automatically guaranteed if the frequencies for all three tile sets are computed from a single dataset. Now, let $\mathcal{M} = \mathcal{T} \cup \mathcal{U} \cup \mathcal{B}$ be the collection containing all tiles. We define the distance between $\mathcal{T}$ and $\mathcal{U}$, w.r.t. $\mathcal{B}$, as

$$d(\mathcal{T}, \mathcal{U}; \mathcal{B}) = \frac{KL(\mathcal{M} \parallel \mathcal{U} \cup \mathcal{B}) + KL(\mathcal{M} \parallel \mathcal{T} \cup \mathcal{B})}{KL(\mathcal{M} \parallel \mathcal{B})} .$$

Using Theorem 2 we can compute the distance in $O(NM)$ time.

If the given tiles are exact, the distance has a simple interpretable form; namely, the distance can be expressed with Jaccard similarity.

Theorem 5. *Assume three tile collections $\mathcal{T}$, $\mathcal{U}$, and $\mathcal{B}$ with exact frequencies $\{\alpha_T\}$, $\{\beta_U\}$, and $\{\gamma_B\}$. Define $X = s(\mathcal{T}) \setminus s(\mathcal{B})$ and $Y = s(\mathcal{U}) \setminus s(\mathcal{B})$. Then $d(\mathcal{T}, \mathcal{U}; \mathcal{B}) = 1 - |X \cap Y| / |X \cup Y|$.*

Example 3. Let us continue Example 2. To compute $d(\mathcal{T}, \mathcal{U}; \emptyset)$ we first note that $\mathcal{T}$ and $\mathcal{U}$ only have exact tiles, and hence we can use Theorem 5. So, we have $|s(\mathcal{T}) \setminus s(\mathcal{U})| = 2$, $|s(\mathcal{U}) \setminus s(\mathcal{T})| = 8$, and $|s(\mathcal{T}) \cup s(\mathcal{U})| = 18$. And hence, the distance $d(\mathcal{T}, \mathcal{U}; \emptyset) = (2 + 8)/18 = 5/9$.

Next, let $\mathcal{M} = \mathcal{T} \cup \mathcal{U} \cup \mathcal{B}$. To compute $d(\mathcal{T}, \mathcal{U}; \mathcal{B})$, note that

$$KL(\mathcal{M} \parallel \mathcal{T} \cup \mathcal{B}) = 2 \log(6) + 10 \log(6/5) \approx 5.4067 \quad .$$

where the first term represents the positive entries in $\mathcal{M}$ and the second term the negative entries in $\mathcal{M}$. Similarly, $KL(\mathcal{M} \parallel \mathcal{B}) \approx 15.2$, and $KL(\mathcal{M} \parallel \mathcal{U} \cup \mathcal{B}) \approx 3.8$. Consequently, the distance is equal to $d(\mathcal{T}, \mathcal{U}; \mathcal{B}) \approx 0.6$ which is slightly larger than $d(\mathcal{T}, \mathcal{U}; \emptyset) \approx 0.56$. This is due to the fact that adding $\mathcal{B}$ to $\mathcal{T}$ makes the probability of encountering a 1 at $(3,4)$ and $(3,5)$ in the model less likely. Hence, given that background knowledge, and regarding $\mathcal{U}$, we are more surprised to find that these entries indeed contain ones.

5 Redescribing Sets of Tiles

Above, we were only concerned in finding out how much information two tile sets share. In this section we consider a more elaborate problem. Namely, given two tile sets, say $\mathcal{T}$ and $\mathcal{U}$, we want to find out which tiles from $\mathcal{U}$ best describe the information provided by $\mathcal{T}$. To this end, we will use our distance as follows.

*Problem 1 (*REDESCRIBE*).* Given three sets of tiles $\mathcal{T}, \mathcal{U}$, and $\mathcal{B}$ with consistent frequencies, find a subset $\mathcal{V} \subseteq \mathcal{U}$ such that $d(\mathcal{V}, \mathcal{T}; \mathcal{B})$ is minimized.

It turns out that finding the best tile subset is computationally intractable.

Theorem 6. *The decision version of* REDESCRIBE *is an **NP**-hard problem.*

Hence, we resort to a simple greedy heuristic: we add iteratively a tile that makes the current tile set closest to the target tile set. We stop the algorithm when we can no longer decrease the distance by adding more tiles.

6 Related Work

To our knowledge, defining a distance between two *general* tile sets is a novel idea. However, there exist several techniques for for comparing datasets using patterns which comes to comparing the *same* pattern set with *different* supports. Such proposals include a Mahalanobis distance between itemset collections [23] and a compression-based distance between itemsets [25]. In addition, Hollmén et al. suggested using L_1 distance between frequent itemset collections, where the missing frequencies were estimated with the support threshold [11].

From technical point of view, comparing pattern sets given background knowledge is akin to defining an interestingness measure based on deviation from the background knowledge. In fact, our approach for building a global Maximum Entropy model from tiles was inspired by the work of De Bie [5], where he builds a similar maximum entropy model from row and column margins (i.e. a Rasch model [21]) and uses it as a static null hypothesis to rank tiles. Further related proposals include iterative mining of patterns by empirical p-values and randomisation [10], and maximum entropy models based on itemsets [27,12].

Several techniques have been proposed for mining sets of tiles. Geerts et al. suggested discovering tilings that cover as many ones as possible [8]. Xiang et al. gave a method to mine (possibly noisy) tiles that cover ones while minimising a cost: the number of transactions and items needed to describe tiles [28]. These methods focus on covering the ones in the data, alternatively, we can assess the quality of a tiling by statistical means. Gionis et al. suggested discovering hierarchical tiles by building a statistical model and optimising an MDL score [9]. De Bie gave a maximum entropy model based on column/row margins to rank tiles [5]. Vreeken et al. propose that the best set of tiles (or itemsets) is the tile set that compresses the dataset best [26].

An alternative approach for discovering tiles is to consider a Boolean matrix factorisation [14]. That is, factorise the dataset into two low rank Boolean matrices, where the row vectors of the one matrix correspond to itemsets, while the column vectors of the other matrix correspond to tid-lists. The Boolean product of these matrices naturally defines a set of noisy tiles.

Compared to tiles, computing maximum entropy models based on itemsets is much more difficult. The reason for this is that there is no equivalent version of Lemma 1 for itemsets. In fact, computing an expected value of an itemset from a maximum entropy model is **PP**-hard [22]. To avoid these problems, we can convert itemsets to exact tiles by considering their supporting transactions.

Redescribing tile sets is closely related to redesciption mining, in which the idea is to find pairs of syntactically different patterns covering roughly the same transactions. Ramakrishnan et al. [20] originally approached the problem by building decision trees. Other approaches include Boolean Formulae with no overlap [7], and exact minimal redescriptions [29]. From a computational point of view, the difference between our problem and existing work is the goal: redescription mining aims to construct an alternative pattern given a *single* target pattern, while we consider *sets* of target and candidate patterns, and aim to find the *subset* of patterns from candidates that together describe the target best.

7 Experiments

In this section we empirically evaluate our measure. We provide our code for research purposes[1]. We evaluate our measure on four publicly available real world datasets. *Abstracts* contains the abstracts of the papers accepted at ICDM up to 2007, where words have been stemmed and stop words removed [5]. The *DNA* amplification data contains information on DNA copy number amplifications. Such copies are known to activate oncogenes and are the hallmarks of nearly all advanced tumours [17]. The *Mammals* presence data consists of presence records of European mammals[2] within geographical areas of 50 × 50 kilometers [15]. Finally, *Paleo* contains information on fossil records[3] found at specific palaeontological sites in Europe [6]. Computing a single distance typically takes

[1] http://www.adrem.ua.ac.be/implementations/

[2] Available for research purposes: http://www.european-mammals.org

[3] NOW public release 030717 available from [6].

Table 1. Number of tiles extracted by each of the considered methods

			Number of Tiles per Method									
Dataset	N	M	*clust*	*bicl*	*atcl*	*sscl*	*asso*	*tiling*	*hyper*	*itt*	*krimp*	*mtv*
Abstracts	859	3933	$5 \times M$	25	753	100	100	38	100	100	100	25
DNA	4590	392	$5 \times M$	25	56	100	100	32	100	100	100	100
Mammals	2183	124	$5 \times M$	25	28	100	91	3	100	2	100	14
Paleo	501	139	$5 \times M$	25	514	100	100	100	100	71	85	14

a few seconds, up to maximally two minutes for the *Abstracts* dataset when considering the most complex sets of tiles and most detailed background knowledge.

7.1 Methods and Mining Results

We apply our measure to compare between the results of ten different exploratory data mining methods for binary data. Table 1 gives an overview, here we state the parameters we use and how we refer to each of the methods between brackets.

We employ simple k-means clustering (*clus*) with $k = 5$ clusters, using L_1 distance. We turn the clusters into tiles by computing column margins inside each cluster. A bi-clustering simultaneously cluster the transactions and the items; a cluster is then a tile defined by the corresponding pair of item and transaction clusters [18]. We apply biclustering (*bicl*) by separately clustering the columns and rows using again k-means clustering ($k = 5$) and combine the two clusterings into a grid. Puolamäki et al. showed that a good approximation bound can be achieved with this approach [19]. Each cluster is represented by a single tile. We use the parameter-free attribute clustering (*atcl*) approach by Mampaey & Vreeken [13], and convert each cluster into a tile — thereby somewhat oversimplifying these results. For subspace clustering (*sscl*), we used the implementation of Müller et al. [16] of the ProClus algorithm [1], mined 100 clusters, each over maximally 32 dimensions, and converted each into a noisy tile. We mined overlapping Tilings [8] of up to 100 exact tiles (*tiling*), allowing the algorithm a maximum of 8 hours. The Asso algorithm [14], (*asso*), was ran with a maximum of 100 factors, of which the non-empty ones were converted into tiles.

Per dataset, we mined up to 100 hyper rectangles [28], (*hyper*), as noisy tiles. We mined Information-Theoretic exact Tiles [5], (*itt*), where the method automatically selects the number of tiles, however, we used top-100 tiles, at most. For mining Maximum-Entropy Tiles (*mtv*) [12], we set a maximum of 2 hours and 100 tiles. We used Krimp [26] to mine itemsets that compress, and took the top-100 most-used itemsets as tiles using their Krimp-usage as *tid*-sets (*krimp*). All four of these methods select itemsets from a candidate collection, for which we used closed frequent itemsets mined at as low as feasible support thresholds, of resp. 5, 1, 800, and 1. For Krimp, however, we could use lower support thresholds for the *Abstract* and *Mammals* datasets, resp. 4 and 300.

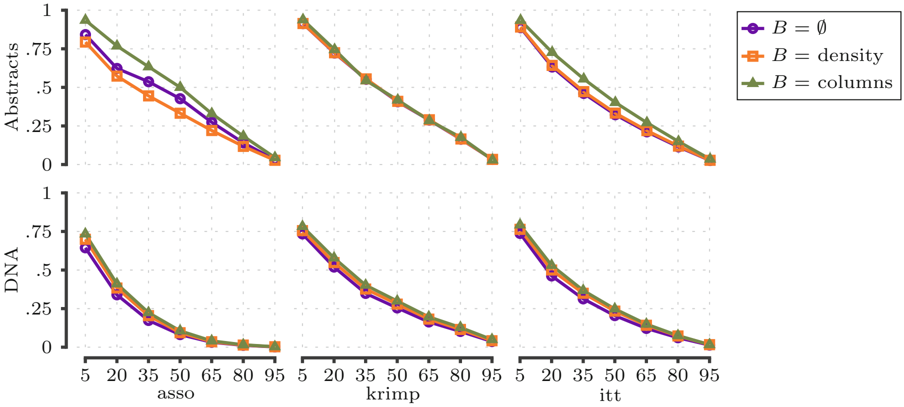


Fig. 2. Distance between top-k tile sets and top-100 tile sets as a function of k. Rows represent datasets while the columns represent the methods.

7.2 Measuring Distances

First, we evaluate whether the measured distance converges to 0 when two sets of tiles approximate each other. To this end, we take the tile sets of *asso*, *krimp*, and *itt*, as obtained on resp. the *Abstracts* and *DNA* datasets. In Figure 2 we plot, per method, the measured distance between the top-k and top-100 tiles. We give the measurements for three different background knowledge settings, resp. no background knowledge, knowledge of the average density of the dataset, and the column margins. The tiles are sorted according to their output order, for *asso* and *itt*, and ascending on code length for *krimp*.

As Figure 2 shows, measurements indeed converge to 0 for higher k, i.e. when the two tile sets become more identical. Adding background information typically increases the distance. This is due to two reasons. First, when density is used, then we can infer additional differences between the areas that are not covered by tiles, thus highlighting the differences. Second, when we are using column margins, we reduce $KL(\mathcal{M} \parallel \mathcal{B})$, the joint information w.r.t. the background knowledge, consequently increasing the distance. Interestingly enough, for *Abstracts* the distances for *asso* decrease when density is used. This is caused by the fact that *asso* produces many overlapping tiles and these overlaps are emphasised when density is used.

7.3 Distances between Results

Our main experiment is to investigate how well we can compare between results of different methods. To do so, for every dataset, and every method considered, we convert their results into tile sets as described above. We measure the pairwise difference between each of these tile sets, using resp. the empty set, overall density, the column margins, and the combination of row and column margins, as background knowledge. For analysis, we present these numerical results, and the

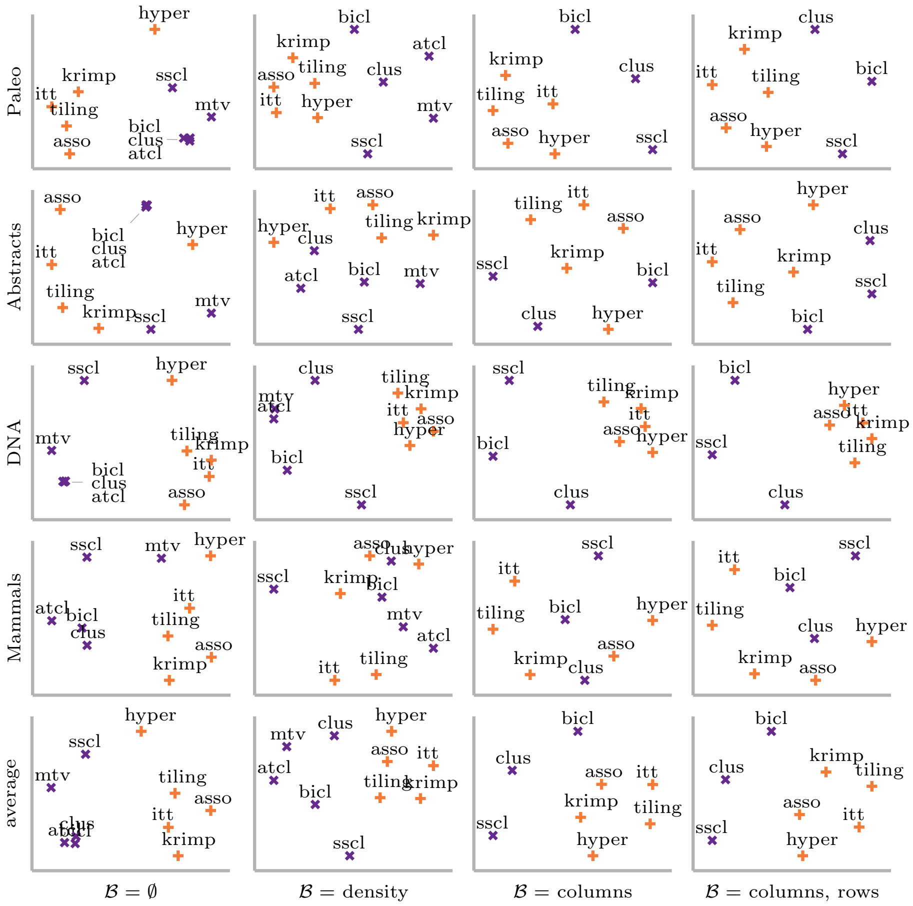


Fig. 3. Sammon projections of distances between tile sets. Each row represents a dataset and each column represents used background knowledge. Note that *atcl* and *mtv* are not included in the rightmost columns, as their tiles provide no information beyond column margins.

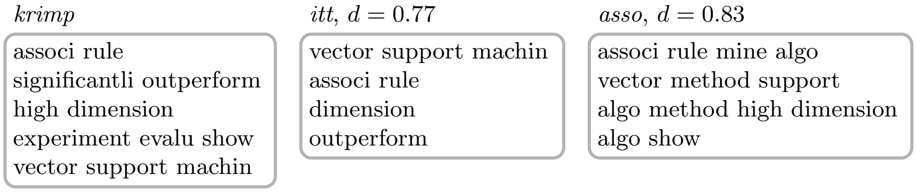
averages over the datasets, visually in Figure 3 by plotting all pairwise distances by Sammon projection. We colour tiling and clustering methods differently.

Considering the first column first, we see that without background knowledge three of the clustering approaches provide virtually the same result. We also see that, albeit not identical, the results of *asso*, *itt*, *krimp*, and *tiling* are relatively close to each other; which makes sense from a conceptual point of view, as these methods are methodologically relatively similar. For *DNA*, the measured dissimilarity between these methods lies between 0.28 and 0.38, whereas the dissimilarities to the other methods measure approximately 0.9.

We observe that *hyper*, while conceptually similar, is measured to provide different results when no background knowledge is given. This is mostly due to

Table 2. Redescribing the results of clustering (*clus*). Measurements for the found redescription, and complete tile set, and the number of selected tiles.

	Redescription / Full Tile Set (# of Tiles)				
Dataset	*asso*	*tiling*	*hyper*	*itt*	*krimp*
Abstracts	.76/.76 (70)	.74/.74 (38)	.50/.57 (32)	.68/.68 (100)	.85/.85 (98)
DNA	.65/.83 (11)	.70/.79 (9)	.67/.84 (21)	.67/.81 (11)	.67/.81 (19)
Mammals	.30/.31 (51)	.68/.68 (3)	.34/.39 (24)	.78/.78 (2)	.49/.49 (91)
Paleo	.68/.83 (16)	.69/.78 (28)	.68/.81 (23)	.73/.81 (21)	.74/.79 (38)



krimp

associ rule
significantli outperform
high dimension
experiment evalu show
vector support machin

itt, $d = 0.77$

vector support machin
associ rule
dimension
outperform

asso, $d = 0.83$

associ rule mine algo
vector method support
algo method high dimension
algo show

Fig. 4. Redescribing 5 selected *krimp* tiles by those discovered by *itt* and *asso*

it including a few very large tiles, that practically cover the whole data, whereas the other methods only cover the data partially. For *hyper* we see that once background knowledge is included, these large tiles are explained away, and the method subsequently becomes part of the 'tiling' group.

Clustering algorithms are close to each other when no background information is used because they all convey the fundamental information of datasets being sparse. When we use density as backgrond knowledge, the differences between clusterings become visible. Interestingly enough, adding row margins to column margins as background information has small impact on the distances.

7.4 Redescribing Results

Next, we empirically evaluate how our measure can be employed with regard to redescribing results; both as validation as well as possible application.

To this end, we first investigate the redescription of results of completely different methods. As such, we take clustering as the target and density as the background information, and redescribe its result by using the tile sets of five of the pattern mining methods as candidate tile sets. Table 2 shows the results of these experiments: for four datasets, the measured divergence between the redescription and the target, the divergence of the complete tile set to the target, and the number of tiles selected for the redescription. First, and foremost, we see that by redescription the measured divergence decreases, which correctly shows that by filtering out, e.g. too specific, tiles that provide information not in the target, we obtain a better description of the target.

We also see, with the exception of *Mammals*, that the measurements are quite high overall, suggesting the clustering provides information these results do not;

not surprising, as these pattern mining methods focus on covering 1s, and not necessarily cover the whole data. Indeed, we see that by providing large and noisy tiles, and so covering more of the data, *asso* and *hyper* lead to the best redescriptions of the clustering results. In particular for *Mammals*, the target can be approximated very well, by resp. only half and a quarter of the total tiles. Overall, we note that typically only fractions of the full tile sets are selected, yet the amount of shared information is larger than for the full tile set: the pattern mining methods provide detailed local information not captured by clustering.

Second, we take a closer look at individual rediscriptions. In order to be able to interpret these, we use the *Abstracts* dataset. We use *asso*, *krimp*, and *itt*, as these provide sufficiently many tiles to choose from; we leave *hyper* out, as for this data it mostly gives only very general tiles, covering all 1s in only 100 tiles.

By hand, we select 5 out of 100 *krimp* tiles, and we identify, for *asso* and *itt*, the sets of tiles that best approximate that partial result, and investigate how well the target concepts are approximated. In Figure 4, we give an example. By the high distances, 0.77 and 0.83, we see the target is not approximated in detail. Overall, for *itt* we find only high-level translations that leave out detail, as its full tile set consists mostly of small itemsets. For *asso*, we see the redescription consists of target tiles combined with general concepts, which together give a reasonable approximation of the target. It is important to note that not simply all intersecting itemsets are given, but only those that provide sufficient information on the target; for both methods, overly large and overly general tiles (e.g. 'high') are not included in the redescription.

8 Discussion

The experimental evaluation of our measure shows it works well in practice, providing insightful groupings of (the results of) exploratory data mining methods on binary data, and meaningful redescriptions of partial results.

The goal of this paper is to take a first step towards comparing between the results of different data mining methods. The method we propose here is for results obtained on binary data. By developing further maximum entropy modelling techniques, however, the same basic idea could be applied to richer data types. We currently straightforwardly convert results into tile sets, capturing much of the information they provide. Ideally, however, we would be able to encode structure beyond simple tile densities, e.g. which attribute-value combinations occur how often (i.e. for *atcl*), such that the information captured in a data mining result can be maintained even more precisely when converted into sets of tiles. Such more complex modelling aside, the general idea of comparing how many possible datasets exhibit such structure remains the same.

Besides measuring divergence of results between different methods, our approach can also be used to choose the most informative result out of many of one randomised method, or, to measure differences between results when varying parameters of a method. It is important to note that our method solely measures the information shared between two sets of tiles; it does not measure the subjective quality of results, nor does it say anything about the ease of analysis of a result. Instead, it gives insight whether or not a, possibly easily interpreted, result is as informative as another, possibly much more complex result.

9 Conclusion

In this paper we discussed comparing results of different explorative data mining algorithms. We argued that any mining result identifies some properties of the data, and that, in an abstract way, we can identify all datasets for which these properties hold. By incorporating these properties into a model using the Maximum Entropy principle, we can measure the shared amount of information by Kullback-Leibler divergence. The measure we construct this way is flexible, and naturally allows including background knowledge, such that differences in results can be measured from the perspective of what a user already knows.

As a first step towards comparing results in general, we formalised our approach for binary data, showed results are easily converted into tiles, and discussed how to incorporate these into a Maximum Entropy model. Our approach provides a means to study and tell differences between results of different data mining methods. As an application, we showed it can be used to parameter-freely identify which parts of results best redescribe a given result. Experiments showed our measure gives meaningful results, correctly identifies methods that are similar in nature, and automatically identifies sound redescriptions of results.

Acknowledgements. The authors wish to thank, in alphabetical order: Tijl De Bie for his implementation of [5]; David Fuhry for his implementation of [28]; Stijn Ligot for running experiments with ProClus [1,16]; Michael Mampaey for the implementations of [13] and [12]; Pauli Miettinen for his implementaiton of Asso [14].

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

References

1. Aggarwal, C.C., Wolf, J.L., Yu, P.S., Procopiuc, C., Park, J.S.: Fast algorithms for projected clustering. In: Proc. ACM SIGMOD 1999, pp. 61–72. ACM, New York (1999)
2. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn (2006)
3. Csiszár, I.: I-divergence geometry of probability distributions and minimization problems. Ann. Prob. 3(1), 146–158 (1975)
4. Darroch, J., Ratcliff, D.: Generalized iterative scaling for log-linear models. Ann. Math. Stat. 43(5), 1470–1480 (1972)
5. De Bie, T.: Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Min. Knowl. Disc. (2010)
6. Fortelius, M., Gionis, A., Jernvall, J., Mannila, H.: Spectral ordering and biochronology of european fossil mammals. Paleobiology 32(2), 206–214 (2006)
7. Gallo, A., Miettinen, P., Mannila, H.: Finding subgroups having several descriptions: Algorithms for redescription mining. In: SDM 2008 (2008)
8. 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)

9. 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)
10. 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: Proc. KDD 2009, pp. 379–388 (2009)
11. Hollmén, J., Seppänen, J.K., Mannila, H.: Mixture models and frequent sets: combining global and local methods for 0-1 data. In: Proc. SDM 2003 (2003)
12. Mampaey, M., Tatti, N., Vreeken, J.: Tell me what I need to know: succinctly summarizing data with itemsets. In: Proc. KDD 2011 (2011)
13. Mampaey, M., Vreeken, J.: Summarising data by clustering items. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010. LNCS, vol. 6322, pp. 321–336. Springer, Heidelberg (2010)
14. Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. IEEE Trans. Knowl. Data Eng. 20(10), 1348–1362 (2008)
15. Mitchell-Jones, A.J., Amori, G., Bogdanowicz, W., Krystufek, B., Reijnders, P.J.H., Spitzenberger, F., Stubbe, M., Thissen, J.B.M., Vohralik, V., Zima, J.: The Atlas of European Mammals. Academic Press, London (1999)
16. Müller, E., Günnemann, S., Assent, I., Seidl, T.: Evaluating clustering in subspace projections of high dimensional data. In: Proc. VLDB 2009 (2009)
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., 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. Puolamäki, K., Hanhijärvi, S., Garriga, G.C.: An approximation ratio for biclustering. Inf. Process. Lett. 108(2), 45–49 (2008)
20. Ramakrishnan, N., Kumar, D., Mishra, B., Potts, M., Helm, R.F.: Turning cartwheels: an alternating algorithm for mining redescriptions. In: Proc. KDD 2004, pp. 266–275 (2004)
21. Rasch, G.: Probabilistic Models for Some Intelligence and Attainnment Tests. Danmarks paedagogiske Institut (1960)
22. Tatti, N.: Computational complexity of queries based on itemsets. Inf. Process. Lett., 183–187 (2006)
23. Tatti, N.: Distances between data sets based on summary statistics. J. Mach. Learn. Res. 8, 131–154 (2007)
24. Tatti, N., Vreeken, J.: Comparing apples and oranges - measuring differences between exploratory data mining results. Technical Report 2011/03, University of Antwerp (2011)
25. Vreeken, J., van Leeuwen, M., Siebes, A.: Characterising the difference. In: Proc. KDD 2007, pp. 765–774 (2007)
26. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: Mining itemsets that compress. Data Min. Knowl. Disc. 23(1), 169–214 (2010)
27. Wang, C., Parthasarathy, S.: Summarizing itemset patterns using probabilistic models. In: Proc. KDD 2006, pp. 730–735 (2006)
28. Xiang, Y., Jin, R., Fuhry, D., Dragan, F.: Summarizing transactional databases with overlapped hyperrectangles. Data Min. Knowl. Disc. (2010)
29. Zaki, M.J., Ramakrishnan, N.: Reasoning about sets using redescription mining. In: Proc. KDD 2005, pp. 364–373 (2005)

Learning Monotone Nonlinear Models Using the Choquet Integral

Ali Fallah Tehrani[1], Weiwei Cheng[1], Krzysztof Dembczyński[1,2],
and Eyke Hüllermeier[1]

[1] Mathematics and Computer Science, University of Marburg, Germany
[2] Institute of Computing Science, Poznań University of Technology, Poland
{fallah,cheng,dembczynski,eyke}@mathematik.uni-marburg.de

Abstract. The learning of predictive models that guarantee monotonicity in the input variables has received increasing attention in machine learning in recent years. While the incorporation of monotonicity constraints is rather simple for certain types of models, it may become a more intricate problem for others. By trend, the difficulty of ensuring monotonicity increases with the flexibility or, say, nonlinearity of a model. In this paper, we advocate the so-called Choquet integral as a tool for learning monotone nonlinear models. While being widely used as a flexible aggregation operator in different fields, such as multiple criteria decision making, the Choquet integral is much less known in machine learning so far. Apart from combining monotonicity and flexibility in a mathematically sound and elegant manner, the Choquet integral has additional features making it attractive from a machine learning point of view. Notably, it offers measures for quantifying the importance of individual predictor variables and the interaction between groups of variables. As a concrete application of the Choquet integral, we propose a generalization of logistic regression. The basic idea of our approach, referred to as choquistic regression, is to replace the linear function of predictor variables, which is commonly used in logistic regression to model the log odds of the positive class, by the Choquet integral.

1 Introduction

A proper specification of the type of dependency between a set of predictor (input) variables $X_1, \ldots, X_m$ and the target (output) variable Y is an important prerequisite for successful model induction. The specification of a corresponding hypothesis space imposes an inductive bias that, amongst others, allows for the incorporation of background knowledge in the learning process. An important type of background knowledge is *monotonicity*: Everything else being equal, the increase (decrease) of a certain input variable X_i can only produce an increase in the output variable Y (e.g., a real number in regression, a class in ordered classification, or the probability of the positive class in binary classification). Adherence to this kind of background knowledge may not only be beneficial for model induction, but is often even considered as a hard constraint. For example,

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 414–429, 2011.

no medical doctor will accept a model in which the probability of cancer is *not* monotone increasing in tobacco consumption.

The simplest type of dependency is a linear relationship:

$$Y = \sum_{i=1}^{m} \alpha_i X_i + \epsilon \; , \tag{1}$$

where $\alpha_1, \ldots, \alpha_m$ are real coefficients and ϵ is an error term. Monotonicity can be guaranteed quite easily for (1), since monotonicity in X_i is equivalent to the constraint $\alpha_i \geq 0$. Another important advantage of (1) is its comprehensibility. In particular, the direction and strength of influence of each predictor X_i are directly reflected by the corresponding coefficient α_i.

Perhaps the sole disadvantage of a linear model is its inflexibility and, coming along with this, the supposed absence of any *interaction* between the variables: The effect of an increase of X_i is always the same, namely $\partial Y / \partial X_i = \alpha_i$, regardless of the values of all other attributes. In many real applications, this assumption is not tenable. Instead, more complex, nonlinear models are needed to properly capture the dependencies between the inputs X_i and the output Y.

An increased flexibility, however, typically comes at the price of a loss in terms of the two previous criteria: comprehensibility is hampered, and monotonicity is more difficult to assure. In fact, as soon as an interaction between attributes is allowed, the influence of an increase in X_i may depend on all other variables, too. As a simple example, consider the extension of (1) by the addition of *interaction terms*, a model which is often used in statistics:

$$Y = \sum_{i=1}^{m} \alpha_i X_i + \sum_{1 \leq i < j \leq m} \alpha_{ij} X_i X_j + \epsilon \; . \tag{2}$$

For this model, $\partial Y / \partial X_i$ is given by $\alpha_i + \sum_{j \neq i} \alpha_{ij} X_j$ and depends on the values of *all* other attributes, which means that, depending on the context as specified by these values, the monotonicity condition may change from one case to another. Consequently, it is difficult to find simple *global* constraints on the coefficients that assure monotonicity. For example, assuming that all attributes are non-negative, it is clear that $\alpha_i \geq 0$ and $\alpha_{ij} \geq 0$ for all $1 \leq i \leq j \leq m$ will imply monotonicity. While being sufficient, however, these constraints are non-necessary conditions, and may therefore impose restrictions on the model space that are more far-ranging than desired; besides, negative interactions cannot be modeled in this way. Quite similar problems occur for commonly used nonlinear methods in machine learning, such as neural networks and kernel machines.

In this paper, we advocate the use of the (discrete) Choquet integral as a tool that is interesting in this regard. As will be argued in more detail later on, the Choquet integral combines the aforementioned properties in a quite convenient and mathematically elegant way: By its very nature as an integral, it is a monotone operator, while at the same time allowing for interactions between attributes. Moreover, the existence of natural measures for quantifying the *importance* of individual and the *interaction* between groups of features, it provides important insights into the model, thereby supporting interpretability.

The rest of this paper is organized as follows. In the next section, we give a brief overview of related work. In Section 3, we recall the basic definition of the Choquet integral and some related notions. In Section 4, we propose a generalization of logistic regression in which the Choquet integral is used to model the log odds of the positive class. Experimental results are presented in Section 5, prior to concluding the paper with a few remarks in Section 6.

2 Related Work

As already mentioned, the problem of monotone classification has received increasing attention in the machine learning community in recent years,[1] despite having been introduced in the literature much earlier [1]. Meanwhile, several machine learning algorithms have been modified so as to guarantee monotonicity in attributes, including nearest neighbor classification [2], neural networks [3], decision tree learning [4,5], rule induction [6], as well as methods based on isotonic regression [7] and piecewise linear models [8].

Instead of modifying learning algorithms so as to guarantee monotone models, another idea is to modify the training data. To this end, data pre-processing methods such as re-labeling techniques have been developed. Such methods seek to repair inconsistencies in the training data, so that (standard) classifiers learned on that data will automatically be monotone [9,10].

Although the Choquet integral has been widely applied as an aggregation operator in multiple criteria decision making [11,12,13], it has been used much less in the field of machine learning so far. There are, however, a few notable exceptions. First, the problem of extracting a Choquet integral (or, more precisely, the non-additive measure on which it is defined) in a data-driven way has been addressed in the literature. Essentially, this is a parameter identification problem, which is commonly formalized as a constraint optimization problem, for example using the sum of squared errors as an objective function [14,15]. To this end, [16] proposed an approach based on the use of quadratic forms, while an alternative heuristic, gradient-based method called HLMS (Heuristic Least Mean Squares) was introduced in [17]. In [18,19], the Choquet integral is used in the context of ordinal classification. Besides, the Choquet integral has been used as an aggregation operator in the context of ensemble learning, i.e., for combining the predictions of different classifiers [20].

3 The Discrete Choquet Integral

In this section, we given a brief introduction to the (discrete) Choquet integral, which, to the best of our knowledge, is not widely known in the field of machine learning so far. Since the Choquet integral can be seen as a generalization of the standard (Lebesque) integral to the case of non-additive measures, we start with a reminder of this type of measure.

[1] For example, a workshop on "Learning Monotone Models from Data" was organized at ECML/PKDD 2009 in Bled, Slovenia.

3.1 Non-additive Measures

Let $C = \{c_1, \ldots, c_m\}$ be a finite set and $\mu : 2^C \to [0,1]$ a measure. For each $A \subseteq C$, we interpret $\mu(A)$ as the *weight* or, say, the *importance* of the set of elements A. As an illustration, one may think of C as a set of criteria (binary features) relevant for a job, like "speaking French" and "programming Java", and of $\mu(A)$ as the evaluation of a candidate satisfying criteria A (and not satisfying $C \backslash A$). The term "criterion" is indeed often used in the decision making literature, where it suggests a monotone "the higher the better" influence.

A standard assumption on a measure $\mu(\cdot)$, which is, for example, at the core of probability theory, is additivity: $\mu(A \cup B) = \mu(A) + \mu(B)$ for all $A, B \subseteq C$ such that $A \cap B = \emptyset$. Unfortunately, additive measures cannot model any kind of interaction between elements: Extending a set of elements A by a set of elements B always increases the weight $\mu(A)$ by the weight $\mu(B)$, regardless of A and B.

Suppose, for example, that the elements of two sets A and B are *complementary* in a certain sense. For instance, $A = \{\texttt{French}, \texttt{Spanish}\}$ and $B = \{\texttt{Java}\}$ could be seen as complementary, since both language skills and programming skills are important for the job. Formally, this can be expressed in terms of a positive interaction: $\mu(A \cup B) > \mu(A) + \mu(B)$. In the extreme case, when language skills and programming skills are indeed essential, $\mu(A \cup B)$ can be high although $\mu(A) = \mu(B) = 0$ (suggesting that a candidate lacking either language or programming skills is completely unacceptable). Likewise, elements can interact in a negative way: If two sets A and B are partly *redundant* or *competitive*, then $\mu(A \cup B) < \mu(A) + \mu(B)$. For example, $A = \{\texttt{C}, \texttt{C\#}\}$ and $B = \{\texttt{Java}\}$ might be seen as redundant, since one programming language does in principle suffice.

The above considerations motivate the use of non-additive measures, also called capacities or fuzzy measures, which are simply normalized and monotone [21]:

$$\mu(\emptyset) = 0, \mu(C) = 1 \quad \text{and} \quad \mu(A) \leq \mu(B) \text{ for all } A \subseteq B \subseteq C \, . \tag{3}$$

A useful representation of non-additive measures, that we shall explore later on for learning Choquet integrals, is in terms of the *Möbius transform*:

$$\mu(B) = \sum_{A \subseteq B} \boldsymbol{m}(A) \tag{4}$$

for all $B \subseteq C$, where the Möbius transform $\boldsymbol{m}_\mu$ of the measure μ is defined as follows:

$$\boldsymbol{m}_\mu(A) = \sum_{B \subseteq A} (-1)^{|A|-|B|} \mu(B) \, . \tag{5}$$

The value $\boldsymbol{m}_\mu(A)$ can be interpreted as the weight that is *exclusively* allocated to A, instead of being indirectly connected with A through the interaction with other subsets.

A measure μ is said to be k-order additive, or simply k-additive, if k is the smallest integer such that $\boldsymbol{m}(A) = 0$ for all $A \subseteq C$ with $|A| > k$. This property is interesting for several reasons. First, as can be seen from (4), it means that a measure μ can formally be specified by significantly fewer than 2^m values, which are needed in the general case. Second, k-additivity is also interesting from a semantic point of view: As will become clear in the following, this property simply means that there are no interaction effects between subsets $A, B \subseteq C$ whose cardinality exceeds k.

3.2 Importance of Criteria and Interaction

An additive (i.e., k-additive with $k = 1$) measure μ can be written as follows:

$$\mu(A) = \sum_{c_i \in A} w_i \ ,$$

with $w_i = \mu(\{c_i\})$ the weight of c_i. Due to (3), these weights are non-negative and such that $\sum_{i=1}^{m} w_i = 1$. In this case, there is obviously no interaction between the criteria c_i, i.e., the influence of a c_i on the value of μ is independent of the presence or absence of any other c_j. Besides, the weight w_i is a natural quantification of the *importance* of c_i.

Measuring the importance of a criterion c_i becomes obviously more involved when μ is non-additive. Besides, one may then also be interested in a measure of *interaction* between the criteria, either pairwise or even of a higher order. In the literature, measures of that kind have been proposed, both for the importance of single as well as the interaction between several criteria.

Given a fuzzy measure μ on C, the *Shaply value* (or importance index) of c_i is defined as a kind of average increase in importance due to adding c_i to another subset $A \subset C$:

$$\varphi(c_i) = \sum_{A \subseteq C \setminus \{c_i\}} \frac{1}{m \begin{pmatrix} m-1 \\ |A| \end{pmatrix}} \Big(\mu(A \cup \{c_i\}) - \mu(A) \Big) \ . \tag{6}$$

The Shaply value of μ is the vector $\varphi(\mu) = (\varphi(c_1), \ldots, \varphi(c_m))$. One can show that $0 \leq \varphi(c_i) \leq 1$ and $\sum_{i=1}^{m} \varphi(c_i) = 1$. Thus, $\varphi(c_i)$ is a measure of the *relative* importance of c_i. Obviously, $\varphi(c_i) = \mu(\{c_i\})$ if μ is additive.

The *interaction index* between criteria c_i and c_j, as proposed by Murofushi and Soneda [22], is defined as follows:

$$I_{i,j} = \sum_{A \subseteq C \setminus \{c_i, c_j\}} \frac{\mu(A \cup \{c_i, c_j\}) - \mu(A \cup \{c_i\}) - \mu(A \cup \{c_j\}) + \mu(A)}{(m-1) \begin{pmatrix} m-2 \\ |A| \end{pmatrix}} \ .$$

This index ranges between -1 and 1 and indicates a positive (negative) interaction between criteria c_i and c_j if $I_{i,j} > 0$ ($I_{i,j} < 0$). The interaction index can also be expressed in terms of the Möbius transform:

$$I_{i,j} = \sum_{K \subseteq C \setminus \{c_i, c_j\}, |K| = k} \frac{1}{k+1} \, \boldsymbol{m}\Big(\{c_i, c_j\} \cup K \Big) \ .$$

Furthermore, as proposed by Grabisch [23], the definition of interaction can be extended to more than two criteria, i.e., to subsets $T \subseteq C$:

$$I_T = \sum_{k=0}^{m-|T|} \frac{1}{k+1} \sum_{K \subseteq C \setminus T, |K|=k} \boldsymbol{m}\Big(T \cup K\Big) \, .$$

3.3 The Choquet Integral

So far, the criteria c_i were simply considered as binary features, which are either present or absent. Mathematically, $\mu(A)$ can thus also be seen as an *integral* of the indicator function of A, namely the function f_A given by $f_A(c) = 1$ if $c \in A$ and $= 0$ otherwise. Now, suppose that $f : C \rightarrow \mathbb{R}_+$ is any non-negative function that assigns a *value* to each criterion c_i; for example, $f(c_i)$ might be the degree to which a candidate satisfies criterion c_i. An important question, then, is how to *aggregate* the evaluations of individual criteria, i.e., the values $f(c_i)$, into an overall evaluation, in which the criteria are properly weighted according to the measure μ. Mathematically, this overall evaluation can be considered as an integral $\mathcal{C}_\mu(f)$ of the function f with respect to the measure μ.

Indeed, if μ is an additive measure, the standard integral just corresponds to the *weighted mean*

$$\mathcal{C}_\mu(f) = \sum_{i=1}^{m} w_i \cdot f(c_i) = \sum_{i=1}^{m} \mu(\{c_i\}) \cdot f(c_i) \, , \tag{7}$$

which is a natural aggregation operator in this case. A non-trivial question, however, is how to generalize (7) in the case where μ is non-additive.

This question, namely how to define the integral of a function with respect to a non-additive measure (not necessarily restricted to the discrete case), is answered in a satisfactory way by the Choquet integral, which has first been proposed for additive measures by Vitali [24] and later on for non-additive measures by Choquet [25]. The point of departure of the Choquet integral is an alternative representation of the "area" under the function f, which, in the additive case, is a natural interpretation of the integral. Roughly speaking, this representation decomposes the area in a "horizontal" instead of a "vertical" manner, thereby making it amenable to a straightforward extension to the non-additive case. More specifically, note that the weighted mean can be expressed as follows:

$$\begin{aligned}\sum_{i=1}^{m} f(c_i) \cdot \mu(\{c_i\}) &= \sum_{i=1}^{m} \Big(f(c_{(i)}) - f(c_{(i-1)})\Big) \cdot \Big(\mu(\{c_{(i)}\}) + \ldots + \mu(\{c_{(n)}\})\Big) \\ &= \sum_{i=1}^{m} \Big(f(c_{(i)}) - f(c_{(i-1)})\Big) \cdot \mu\Big(A_{(i)}\Big) \, ,\end{aligned}$$

where $(\cdot)$ is a permutation of $\{1, \ldots, m\}$ such that $0 \leq f(c_{(1)}) \leq f(c_{(2)}) \leq \ldots \leq f(c_{(m)})$ (and $f(c_{(0)}) = 0$ by definition), and $A_{(i)} = \{c_{(i)}, \ldots, c_{(m)}\}$; see Fig. 1 as an illustration.

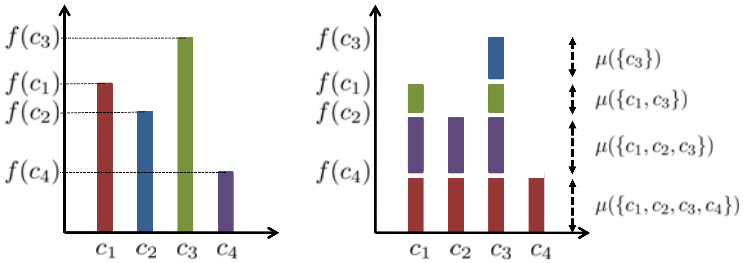


Fig. 1. Vertical (left) versus horizontal (right) integration. In the first case, the height of a single bar, $f(c_i)$, is multiplied with its "width" (the weight $\mu(\{c_i\})$), and these products are added. In the second case, the height of each horizontal section, $f(c_{(i)}) - f(c_{(i-1)})$, is multiplied with the corresponding "width" $\mu(A_{(i)})$.

Now, the key difference between the left and right-hand side of the above expression is that, whereas the measure μ is only evaluated on single elements c_i on the left, it is evaluated on *subsets* of elements on the right. Thus, the right-hand side suggests an immediate extension to the case of non-additive measures, namely the Choquet integral, which, in the discrete case, is formally defined as follows:

$$\mathcal{C}_\mu(f) = \sum_{i=1}^{m} \left(f(c_{(i)}) - f(c_{(i-1)})\right) \cdot \mu(A_{(i)})$$

In terms of the Möbius transform of μ, the Choquet integral can also be expressed as follows:

$$\begin{aligned} \mathcal{C}_\mu(f) &= \sum_{i=1}^{m} \left(f(c_{(i)}) - f(c_{(i-1)})\right) \cdot \mu(A_{(i)}) \\ &= \sum_{i=1}^{m} f(c_{(i)}) \cdot (\mu(A_{(i)}) - \mu(A_{(i+1)})) \\ &= \sum_{i=1}^{m} f(c_{(i)}) \sum_{R \subseteq T_{(i)}} \boldsymbol{m}(R) \\ &= \sum_{T \subseteq C} \boldsymbol{m}(T) \times \min_{i \in T} f(c_i) \end{aligned} \tag{8}$$

where $T_{(i)} = \left\{S \cup \{c_{(i)}\} \,|\, S \subset \{c_{(i+1)}, \ldots, c_{(m)}\}\right\}$.

4 Choquistic Regression

Consider the standard setting of binary classification, where the goal is to predict the value of an output (response) variable $y \in \mathcal{Y} = \{0, 1\}$ for a given instance

$$\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_m$$

represented in terms of a feature vector. More specifically, the goal is to learn a classifier $\mathcal{L} : \mathcal{X} \rightarrow \mathcal{Y}$ from a given set of (i.i.d.) training data

$$\mathcal{D} = \left\{ (\boldsymbol{x}^{(i)}, y^{(i)}) \right\}_{i=1}^{n} \subset (\mathcal{X} \times \mathcal{Y})^n$$

so as to minimize the risk

$$R(\mathcal{L}) = \int_{\mathcal{X} \times \mathcal{Y}} \ell(\mathcal{L}(\boldsymbol{x}), y) \, d\mathbf{P}_{XY}(\boldsymbol{x}, y) \ ,$$

where $\ell(\cdot)$ is a loss function (e.g., the simple 0/1 loss given by $\ell(\hat{y}, y) = 0$ if $\hat{y} = y$ and $= 1$ if $\hat{y} \neq y$).

Logistic regression is a well-established statistical method for (probabilistic) classification [26]. Its popularity is due to a number of appealing properties, including monotonicity and comprehensibility: Since the model is essentially *linear* in the input attributes, the strength of influence of each predictor is directly reflected by the corresponding regression coefficient. Moreover, the influence of each attribute is *monotone* in the sense that an increase of the value of the attribute can only increase (decrease) the probability of the positive class.

Formally, the probability of the positive class (and hence of the negative class) is modeled as a generalized linear function of the input attributes, namely in terms of the logarithm of the probability ratio:

$$\log \left(\frac{\mathbf{P}(y = 1 \,|\, \boldsymbol{x})}{\mathbf{P}(y = 0 \,|\, \boldsymbol{x})} \right) = w_0 + \boldsymbol{w}^\top \boldsymbol{x} \ , \tag{9}$$

where $\boldsymbol{w} = (w_1, w_2, \ldots, w_m) \in \mathbb{R}^m$ is a vector of regression coefficients and $w_0 \in \mathbb{R}$ a constant bias (the intercept). A positive regression coefficient $w_i > 0$ means that an increase of the predictor variable x_i will increase the probability of a positive response, while a negative coefficient implies a decrease of this probability. Besides, the larger the absolute value $|w_i|$ of the regression coefficient, the stronger the influence of x_i.

Since $\mathbf{P}(y = 0 \,|\, \boldsymbol{x}) = 1 - \mathbf{P}(y = 1 \,|\, \boldsymbol{x})$, a simple calculation yields the posterior probability

$$\pi_l \overset{\mathrm{df}}{=} \mathbf{P}(y = 1 \,|\, \boldsymbol{x}) = \Big(1 + \exp(-w_0 - \boldsymbol{w}^\top \boldsymbol{x}) \Big)^{-1} \ . \tag{10}$$

The logistic function $z \mapsto (1 + \exp(-z))^{-1}$, which has a sigmoidal shape, is a specific type of *link function*.

Needless to say, the linearity of the above model is a strong restriction from a learning point of view, and the possibility of interactions between predictor variables has of course also been noticed in the statistical literature [27]. A standard way to handle such interaction effects is to add interaction terms to the linear function of predictor variables, like in (2). As explained earlier, however, the aforementioned advantages of logistic regression will then be lost.

In the following, we therefore propose an extension of logistic regression that allows for modeling nonlinear relationships between input and output variables while preserving the advantages of comprehensibility and monotonicity.

4.1 The Choquistic Model

In order to model nonlinear dependencies between predictor variables and response, and to take interactions between predictors into account, we propose to extend the logistic regression model by replacing the linear function $\boldsymbol{x} \mapsto w_0 + \boldsymbol{w}^\top \boldsymbol{x}$ in (9) by the Choquet integral. More specifically, we propose the following model

$$\pi_c \overset{\text{df}}{=} \mathbf{P}(y = 1 \,|\, \boldsymbol{x}) = \Big(1 + \exp(-\gamma\,(\mathcal{C}_\mu(f_{\boldsymbol{x}}) - \beta))\Big)^{-1}, \tag{11}$$

where $\mathcal{C}_\mu(f_{\boldsymbol{x}})$ is the Choquet integral (with respect to the measure μ) of the function $f_{\boldsymbol{x}} : \{c_1, \ldots, c_m\} \to [0,1]$ that maps each attribute c_i to a normalized value $x_i = f_{\boldsymbol{x}}(c_i) \in [0,1]$; $\beta, \gamma \in \mathbb{R}$ are constants.

The normalization is meant to turn each predictor variable into a criterion, i.e., a “the higher the better” attribute, and to assure commensurability between the criteria [28]. A simple transformation, that we shall also employ in our experimental studies, is given by the mapping $z_i = (x_i - m_i)/(M_i - m_i)$, where m_i and M_i are lower and upper bounds for x_i (perhaps estimated from the data); if the influence of x_i is actually negative (i.e., $w_i < 0$), then the mapping $z_i = (M_i - x_i)/(M_i - m_i)$ is used instead.

In order to verify that our model (11) is a proper generalization of standard logistic regression, recall that the Choquet integral reduces to a weighted mean (7) in the special case of an additive measure μ. Moreover, consider any linear function $\boldsymbol{x} \mapsto g(\boldsymbol{x}) = w_0 + \boldsymbol{w}^\top \boldsymbol{x}$ with $\boldsymbol{w} = (w_1, \ldots, w_m)$. This function can also be written in the form

$$\begin{aligned} g(\boldsymbol{x}) &= w_0 + \sum_{i=1}^m \left(w_i p_i + |w_i|(M_i - m_i) z_i\right) \\ &= w_0 + \sum_{i=1}^m w_i p_i + \sum_{i=1}^m |w_i|(M_i - m_i) z_i \\ &= w_0' + \left(\sum_{i=1}^m u_i\right)^{-1} \sum_{i=1}^m u_i' z_i \\ &= \gamma \left(\sum_{i=1}^m u_i' z_i - \beta\right), \end{aligned}$$

where $p_i = m_i$ if $w_i \geq 0$ and $p_i = M_i$ if $w_i < 0$, $u_i = |w_i|(M_i - m_i)$, $\gamma = \left(\sum_{i=1}^m u_i\right)^{-1}$, $u_i' = u_i/\gamma$, $w_0' = w_0 + \sum_{i=1}^m w_i p_i$, $\beta = -w_0'/\gamma$. By definition, the u_i' are non-negative and sum up to 1, which means that $\sum_{i=1}^m u_i' z_i$ is a weighted mean of the z_i that can be represented by a Choquet integral.

Recalling the idea of “evaluating” an instance $\boldsymbol{x}$ in terms of a set of criteria, the model (11) can be seen as a two-step procedure: The first step consists of an assessment of $\boldsymbol{x}$ in terms of a (latent) utility degree

$$u = U(\boldsymbol{x}) = \mathcal{C}_\mu(f_{\boldsymbol{x}}) \in [0,1].$$

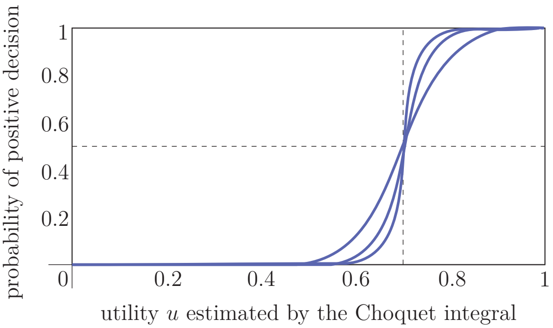


Fig. 2. Probability of a positive decision, $\mathbf{P}(y = 1 \,|\, \boldsymbol{x})$, as a function of the estimated degree of utility, $u = U(\boldsymbol{x})$, for a threshold $\beta = 0.7$ and different values of γ

Then, in a second step, a discrete choice (yes/no decision) is made on the basis of this utility. Roughly speaking, this is done through a "probabilistic thresholding" at the utility threshold β. If $U(\boldsymbol{x}) > \beta$, then the decision tends to be positive, whereas if $U(\boldsymbol{x}) < \beta$, it tends to be negative. The precision of this decision is determined by the parameter γ (see Fig. 2): For large γ, the decision function converges toward the step function $u \mapsto \mathbb{I}(u > \beta)$, jumping from 0 to 1 at β. For small γ, this function is smooth, and there is a certain probability to violate the threshold rule $u \mapsto \mathbb{I}(u > \beta)$. This might be due to the fact that, despite being important for decision making, some properties of the instances to be classified are not captured by the utility function. In that case, the utility $U(\boldsymbol{x})$, estimated on the basis of the given attributes, is not a perfect predictor for the decision eventually made. Thus, the parameter γ can also be seen as an indicator of the quality of the classification model.

4.2 Parameter Estimation

The model (11) has several degrees of freedom: The fuzzy measure μ (Möbius transform $\boldsymbol{m} = \boldsymbol{m}_\mu$) determines the (latent) utility function, while the utility threshold β and the scaling parameter γ determine the discrete choice model. The goal of learning is to identify these degrees of freedom on the basis of the training data $\mathcal{D}$. Like in the case of standard logistic regression, it is possible to harness the maximum likelihood (ML) principle for this purpose. The log-likelihood of the parameters can be written as

$$\begin{aligned} l(\boldsymbol{m}, \gamma, \beta) &= \log \mathbf{P}(\mathcal{D} \,|\, \boldsymbol{m}, \beta, \gamma) \\ &= \log \left(\prod_{i=1}^{n} \mathbf{P}(y^{(i)} \,|\, \boldsymbol{x}^{(i)}; \boldsymbol{m}, \beta, \gamma) \right) \qquad (12) \\ &= \sum_{i=1}^{n} y^{(i)} \log \pi_c^{(i)} + \left(1 - y^{(i)}\right) \log \left(1 - \pi_c^{(i)}\right) . \end{aligned}$$

One easily verifies that (12) is convex with respect to $\boldsymbol{m}, \gamma$, and β. In principle, maximization of the log-likelihood can be accomplished by means of standard gradient-based optimization methods. However, since we have to assure that μ is a proper fuzzy measure and, hence, that $\boldsymbol{m}$ guarantees the corresponding monotonicity and boundary conditions, we actually need to solve a *constrained* optimization problem:

$$\max_{\boldsymbol{m},\gamma,\beta} \left\{ -\gamma \sum_{i=1}^{n} (1-y^{(i)})(\mathcal{C}_{\boldsymbol{m}}(\boldsymbol{x}^{(i)}) - \beta) - \sum_{i=1}^{n} \log \Big(1 + \exp(-\gamma \left(\mathcal{C}_{\boldsymbol{m}}(\boldsymbol{x}^{(i)}) - \beta\right)) \Big) \right\}$$

$$\text{s.t.} \quad \gamma > 0, \; 0 \leq \beta \leq 1, \; \sum_{T \subseteq C} \boldsymbol{m}(T) = 1, \quad \text{and}$$

$$\sum_{B \subseteq A \setminus \{c_i\}} \boldsymbol{m}(B \cup \{c_i\}) \geq 0 \quad \forall A \subseteq C, \forall c_i \in C.$$

A solution to this problem can be produced by standard solvers. Concretely, we used the `fmincon` function implemented in the optimization toolbox of Matlab. This method is based on a sequential quadratic programming approach.

Recall that, once the model has been identified, the importance of each attribute and the degree of interaction between groups of attributes can be derived from the Möbius transform $\boldsymbol{m}$; these are given, respectively, by the Shapley value and the interaction indexes as introduced in Section 3.2.

5 Experiments

5.1 Data Sets

Although the topic is receiving increasing interest in the machine learning community, benchmark data for monotone classification is by far not as abundant as for conventional classification. In total, we managed to collect 9 data sets from different sources, notably the UCI repository[2] and the WEKA machine learning framework [29], for which monotonicity in the input variables is a reasonable assumption; see Table 1 for a summary. All the data sets can be downloaded at our website[3]. Many of them have also been used in previous studies on monotone learning. Some of them have a numerical or ordered categorical output, however. These outputs were binarized by thresholding at the median. Moreover, all input attributes were normalized.

5.2 Methods

Since choquistic regression (CR) can be seen as an extension of standard logistic regression (LR), it is natural to compare these two methods. Essentially,

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

[3] `http://www.uni-marburg.de/fb12/kebi/research/`

Table 1. Data sets and their properties

data set	#instances	#attributes	source
Den Bosch (DBS)	120	8	[30]
CPU	209	6	UCI
Breast Cancer (BCC)	286	9	UCI
Auto MPG	392	7	UCI
Employee Selection (ESL)	488	4	WEKA
Mammographic (MMG)	961	6	UCI
Employee Rejection/Acceptance (ERA)	1000	4	WEKA
Lecturers Evaluation (LEV)	1000	4	WEKA
Car Evaluation (CEV)	1728	6	UCI

this comparison should give an idea of the usefulness of an increased flexibility. On the other side, one may also ask for the usefulness of assuring monotonicity. Therefore, we additionally included two other extensions of LR, which are flexible but not necessarily monotone, namely kernel logistic regression (KLR) with polynomial and Gaussian kernels. The degree of the polynomial kernel was set to 2, so that it models low-level interactions of the features. The Gaussian kernel, on the other hand, is able to capture interactions of higher order. For each data set, the width parameter of the Gaussian kernel was selected from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ in the most favorable way. Finally, we included a method which is both monotone and flexible, namely the MORE algorithm for learning rule ensembles under monotonicity constraints [6].

5.3 Results

Classification accuracy was measured in terms of 0/1 loss and determined by randomly splitting the data into two parts, one part for training and one part for testing. This was repeated 100 times, and the accuracy degrees were averaged.

A possible improvement of CR over its competitors, in terms of predictive accuracy, may be due to two reasons: First, in comparison to standard LR, it is more flexible and has the ability to capture nonlinear dependencies between input attributes. Second, in comparison to non-monotone learners, it takes background knowledge about the dependency between input and output variables into consideration.

Both aspects have to be put in perspective, however. First, regarding flexibility, it is clear that an improvement is unlikely unless additional flexibility is indeed needed. On the contrary, if the true underlying dependency is indeed a linear one, at least approximately, then standard logistic regression will be the model of choice, whereas CR may tend to overfit the training data and hence generalize worse. Regarding monotonicity, previous studies have indeed shown that improvements are possible, albeit of a small margin. In fact, upon closer examination, the benefit of enforcing monotonicity is not entirely obvious [31]. Moreover, the more extensive the training data, the smaller the improvement

Table 2. Classification performance in terms of the mean and standard deviation of 0/1 loss. From top to bottom: 20%, 50%, and 80% training data.

dataset	CR	LR	KLR-ply	KLR-rbf	MORE
DBS	.2226±.0380 (4)	.1803±.0336 (1)	.2067±.0447 (3)	.1922±.0501 (2)	.2541±.0142 (5)
CPU	.0457±.0338 (2)	.0430±.0318 (1)	.0586±.0203 (3)	.0674±.0276 (4)	.1033±.0681 (5)
BCC	.2939±.0100 (4)	.2761±.0265 (1)	.3102±.0386 (5)	.2859±.0329 (3)	.2781±.0219 (2)
MPG	.0688±.0098 (2)	.0664±.0162 (1)	.0729±.0116 (4)	.0705±.0122 (3)	.0800±.0198 (5)
ESL	.0764±.0291 (3)	.0747±.0243 (1)	.0752±.0117 (2)	.0794±.0134 (4)	.1035±.0332 (5)
MMG	.1816±.0140 (3)	.1752±.0106 (2)	.1970±.0095 (4)	.2011±.0123 (5)	.1670±.0120 (1)
ERA	.2997±.0123 (2)	.2922±.0096 (1)	.3011±.0132 (3)	.3259±.0172 (5)	.3040±.0192 (4)
LEV	.1527±.0138 (1)	.1644±.0106 (4)	.1570±.0116 (2)	.1577±.0124 (3)	.1878±.0242 (5)
CEV	.0441±.0128 (1)	.1689±.0066 (5)	.0571±.0078 (3)	.0522±.0085 (2)	.0690±.0408 (4)
avg. rank	2.4	1.9	3.3	3.4	4
DBS	.1560±.0405 (3)	.1443±.0371 (2)	.1845±.0347 (5)	.1628±.0269 (4)	.1358±.0432 (1)
CPU	.0156±.0135 (1)	.0400±.0106 (3)	.0377±.0153 (2)	.0442±.0223 (5)	.0417±.0198 (4)
BCC	.2871±.0358 (4)	.2647±.0267 (2)	.2706±.0295 (3)	.2879±.0269 (5)	.2616±.0320 (1)
MPG	.0641±.0175 (1)	.0684±.0206 (2)	.1462±.0218 (5)	.1361±.0197 (4)	.0700±.0162 (3)
ESL	.0660±.0135 (1)	.0697±.0144 (3)	.0704±.0128 (5)	.0699±.0148 (4)	.0690±.0171 (2)
MMG	.1736±.0157 (3)	.1710±.0161 (2)	.1859±.0141 (4)	.1900±.0169 (5)	.1604±.0139 (1)
ERA	.3008±.0135 (3)	.3054±.0140 (4)	.2907±.0136 (1)	.3084±.0152 (5)	.2928±.0168 (2)
LEV	.1357±.0122 (1)	.1641±.0131 (4)	.1500±.0098 (3)	.1482±.0112 (2)	.1658±.0202 (5)
CEV	.0346±.0076 (1)	.1667±.0093 (5)	.0357±.0113 (2)	.0393±.0090 (3)	.0443±.0080 (4)
avg. rank	2	3	3.3	4.1	2.6
DBS	.1363±.0380 (2)	.1409±.0336 (4)	.1422±.0498 (5)	.1386±.0521 (3)	.0974±.0560 (1)
CPU	.0089±.0126 (1)	.0366±.0068 (4)	.0329±.0295 (2)	.0384±.0326 (5)	.0342±.0232 (3)
BCC	.2631±.0424 (2)	.2669±.0483 (3)	.2784±.0277 (4)	.2937±.0297 (5)	.2526±.0472 (1)
MPG	.0526±.0263 (1)	.0538±.0282 (2)	.0669±.0251 (4)	.0814±.0309 (5)	.0656±.0248 (3)
ESL	.0517±.0235 (1)	.0602±.0264 (2)	.0654±.0228 (3)	.0718±.0188 (5)	.0657±.0251 (4)
MMG	.1584±.0255 (2)	.1683±.0231 (3)	.1798±.0293 (4)	.1853±.0232 (5)	.1521±.0249 (1)
ERA	.2855±.0257 (1)	.2932±.0261 (4)	.2885±.0302 (2)	.2951±.0286 (5)	.2894±.0278 (3)
LEV	.1312±.0186 (1)	.1662±.0171 (5)	.1518±.0104 (3)	.1390±.0129 (2)	.1562±.0252 (4)
CEV	.0221±.0091 (1)	.1643±.0184 (5)	.0376±.0091 (3)	.0262±.0067 (2)	.0408±.0090 (4)
avg. rank	1.3	3.6	3.3	4.1	2.7

tends to be. This is understandable, since background knowledge will lose importance with an increasing number of observations.

The results of the experiments are summarized in Table 2 and 3. As can be seen, CR compares quite favorably with the other approaches, especially with the non-monotone KLR methods. It also outperforms LR, at least for sufficiently extensive training data; if the amount of training data is small, however, LR is even better, probably because CR will then tend to overfit the data. Finally, CR also compares favorably with MORE, although the difference in terms of the average ranks is not statistically significant (the critical distance for the Nemenyi test at significance level 0.05 is 2.03).

In Fig. 3, a visualization of the (pairwise) interaction between attributes is shown for the car evaluation data, for which CR performs significantly better than LR. In this data set, the evaluation of a car (output attribute) depends on a number of criteria, namely (a) buying price, (b) price of the maintenance, (c) number of doors, (d) capacity in terms of persons to carry, (e) size of luggage boot, (f) safety of the car. These criteria form a natural hierarchy: (a) and (b) form a subgroup PRICE, whereas the other properties are of a TECHNICAL nature and can be further decomposed into COMFORT (c–e) and safety (f). Interestingly, the interaction in our model nicely agrees with this hierarchy: Interaction within each subgroup tends to be smaller (as can be seen from the

Table 3. Win statistics (number of data sets on which the first method was better than the second one) for 20%, 50%, and 80% training data

	CR	LR	KLR-ply	KLR-rbf	MORE
CR	–	2 | 6 | 9	7 | 7 | 9	7 | 9 | 9	7 | 5 | 6
LR	7 | 3 | 0	–	7 | 5 | 5	7 | 7 | 6	7 | 3 | 2
KLR-ply	2 | 2 | 0	2 | 4 | 4	–	5 | 5 | 6	7 | 4 | 5
KLR-rbf	2 | 0 | 0	2 | 2 | 3	4 | 4 | 3	–	6 | 2 | 2
MORE	2 | 4 | 3	2 | 6 | 7	2 | 5 | 4	3 | 7 | 7	–

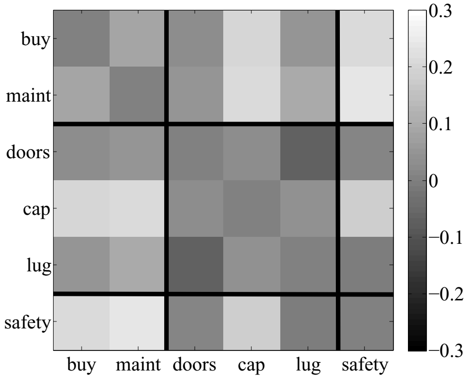


Fig. 3. Visualization of the interaction index for the car evaluation data (numerical values are shown in terms of level of gray, values on the diagonal are set to 0). Groups of related criteria are indicated by the black lines.

darker colors) than interaction between criteria from different subgroups, suggesting a kind of redundancy in the former and complementarity in the latter case.

6 Concluding Remarks

In this paper, we have advocated the use of the discrete Choquet integral as an aggregation operator in machine learning, especially in the context of learning monotone models. Apart from combining monotonicity and flexibility in a mathematically sound and elegant manner, the Choquet integral offers measures for quantifying the importance of individual predictor variables and the interaction between groups of variables, thereby providing important information about the relationship between independent and dependent variables.

As a concrete application, we have proposed a generalization of logistic regression, in which the Choquet integral is used for modeling the log odds of the positive class. First experimental studies have shown that this method, called choquistic regression, compares quite favorably with other methods. We like to mention again, however, that an improvement in prediction accuracy should not necessarily be seen as the main goal of monotone learning. Instead, the adherence to monotonicity constraints is often an important prerequisite for the acceptance of a model by domain experts.

An interesting question to be addressed in future work concerns a possible restriction of the choquistic model to k-additive measures, for a suitable value of k. This may have two important advantages: First, it may prevent from overfitting the data in cases where the full flexibility of the Choquet integral is actually not needed. Second, since less parameters need to be identified, the computational complexity will be reduced, too. Of course, the key problem to be addressed in this regard concerns the question of how to choose k in the most favorable way.

Beyond that, the Choquet integral can of course be combined with other machine learning methods, and its use is not restricted to (binary) classification. We are quite convinced of its high potential in machine learning in general, and we are looking forward to exploring this potential in greater detail.

References

1. Ben-David, A., Sterling, L., Pao, Y.-H.: Learning and classification of monotonic ordinal concepts. Computational Intelligence 5(1), 45–49 (1989)
2. Duivesteijn, W., Feelders, A.: Nearest neighbour classification with monotonicity constraints. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 301–316. Springer, Heidelberg (2008)
3. Sill, J.: Monotonic networks. In: Advances in Neural Information Processing Systems, pp. 661–667. The MIT Press, Denver (1998)
4. Ben-David, A.: Monotonicity maintenance in information-theoretic machine learning algorithms. Machine Learning 19, 29–43 (1995)
5. Potharst, R., Feelders, A.: Classification trees for problems with monotonicity constraints. ACM SIGKDD Explorations Newsletter 4(1), 1–10 (2002)
6. Dembczyński, K., Kotlowski, W., Slowinski, R.: Learning rule ensembles for ordinal classification with monotonicity constraints. Fundamenta Informaticae 94(2), 163–178 (2009)
7. Chandrasekaran, R., Ryu, Y., Jacob, V., Hong, S.: Isotonic separation. INFORMS Journal on Computing 17, 462–474 (2005)
8. Dembczyński, K., Kotłowski, W., Słowiński, R.: Additive preference model with piecewise linear components resulting from dominance-based rough set approximations. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 499–508. Springer, Heidelberg (2006)
9. Feelders, A.: Monotone relabeling in ordinal classification. In: Proceedings of the 10th IEEE International Conference on Data Mining, pp. 803–808. IEEE Computer Society, Los Alamitos (2010)
10. Kotłowski, W., Dembczyński, K., Greco, S., Słowiński, R.: Stochastic dominance-based rough set model for ordinal classification. Information Sciences 178(21), 3989–4204 (2008)
11. Grabisch, M., Murofushi, T., Sugeno, M. (eds.): Fuzzy Measures and Integrals: Theory and Applications. Physica, Heidelberg (2000)

12. Grabisch, M.: Fuzzy integral in multicriteria decision making. Fuzzy Sets and Systems 69(3), 279–298 (1995)
13. Torra, V.: Learning aggregation operators for preference modeling. In: Preference Learning, pp. 317–333. Springer, Heidelberg (2011)
14. Torra, V., Narukawa, Y.: Modeling Decisions: Information Fusion and Aggregation Operators. Springer, Heidelberg (2007)
15. Grabisch, M.: Modelling data by the Choquet integral. In: Information Fusion in Data Mining, pp. 135–148. Springer, Heidelberg (2003)
16. Mori, T., Murofushi, T.: An analysis of evaluation model using fuzzy measure and the Choquet integral. In: Proceedings of the 5th Fuzzy System Symposium, pp. 207–212. Japan Society for Fuzzy Sets and Systems (1989)
17. Grabisch, M.: A new algorithm for identifying fuzzy measures and its application to pattern recognition. In: Proceedings of IEEE International Conference on Fuzzy Systems, vol. 1, pp. 145–150. IEEE, Los Alamitos (1995)
18. Angilella, S., Greco, S., Matarazzo, B.: Non-additive robust ordinal regression with Choquet integral, bipolar and level dependent Choquet integrals. In: Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference, IFSA/EUSFLAT, pp. 1194–1199 (2009)
19. Beliakov, G., James, S.: Citation-based journal ranks: the use of fuzzy measures. Fuzzy Sets and Systems 167(1), 101–119 (2011)
20. Grabisch, M., Nicolas, J.-M.: Classification by fuzzy integral: performance and tests. Fuzzy Sets and Systems 65(2-3), 255–271 (1994)
21. Sugeno, M.: Theory of Fuzzy Integrals and its Application. PhD thesis, Tokyo Institute of Technology (1974)
22. Murofushi, T., Soneda, S.: Techniques for reading fuzzy measures (III): interaction index. In: Proceedings of the 9th Fuzzy Systems Symposium, pp. 693–696 (1993)
23. Grabisch, M.: k-order additive discrete fuzzy measures and their representation. Fuzzy Sets and Systems 92(2), 167–189 (1997)
24. Vitali, G.: Sulla definizione di integrale delle funzioni di una variabile. Annali di Matematica Pura ed Applicata 2(1), 111–121 (1925)
25. Choquet, G.: Theory of capacities. Annales de l'institut Fourier 5, 131–295 (1954)
26. Hosmer, D., Lemeshow, S.: Applied Logistic Regression, 2nd edn. Wiley, Chichester (2000)
27. Jaccard, J.: Interaction Effects in Logistic Regression. Saga Publications, Thousand Oaks (2001)
28. Modave, F., Grabisch, M.: Preference representation by a Choquet integral: commensurability hypothesis. In: Proceedings of the 7th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pp. 164–171. Editions EDK (1998)
29. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
30. Daniels, H., Kamp, B.: Applications of mlp networks to bond rating and house pricing. Neural Computation and Applications 8, 226–234 (1999)
31. Ben-David, A., Sterling, L., Tran, T.: Adding monotonicity to learning algorithms impair their accuracy. Expert Systems with Applications 36(3), 6627–6634 (2009)

Feature Selection for Transfer Learning

Selen Uguroglu and Jaime Carbonell

Language Technologies Institute, Carnegie Mellon University
{sugurogl,jgc}@cs.cmu.edu

Abstract. Common assumption in most machine learning algorithms is that, labeled (source) data and unlabeled (target) data are sampled from the same distribution. However, many real world tasks violate this assumption: in temporal domains, feature distributions may vary over time, clinical studies may have sampling bias, or sometimes sufficient labeled data for the domain of interest does not exist, and labeled data from a related domain must be utilized. In such settings, knowing in which dimensions source and target data vary is extremely important to reduce the distance between domains and accurately transfer knowledge. In this paper, we present a novel method to identify variant and invariant features between two datasets. Our contribution is two fold: First, we present a novel transfer learning approach for domain adaptation, and second, we formalize the problem of finding differently distributed features as a convex optimization problem. Experimental studies on synthetic and benchmark real world datasets show that our approach outperform other transfer learning approaches, and it aids the prediction accuracy significantly.

1 Introduction

In real life applications of supervised machine learning, the conditions in which the models are developed and used may differ. For instance, in clinical studies of drugs, the selection of patients may not be representative of the general population: the sample may have a gender bias, race bias, or patients in the study may have a lower health status. A network intrusion software or a spam detection software developed many years ago may not be predictive anymore, due to newer attack or spam patterns. A survey conducted in a region, may not be applicable to another region, due to differences in the populations. These examples conflict with the major assumption of machine learning, that training and testing data come from the same distribution. Moreover it is not always guaranteed that there is sufficient labeled data in the target (newer) domain to train a new model.

There have been many efforts to deal with such situations, including relatively new learning paradigms, such as transfer learning. Transfer learning tries to utilize the readily available labeled data from another domain for prediction in the target domain of interest. This approach is also known as domain adaptation. An example application area for domain adaptation is sentiment analysis, where one intends to use reviews in a particular domain, say stock reviews, to predict the sentiments in another domain, say computer reviews. How products

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 430–442, 2011.

are described in the reviews differ across domains, and therefore two dataset distributions may be different [10]. In bioinformatics, one may want to utilize labeled clinical data from one institution, to predict high-risk patients in another institution. Although the two clinical datasets may share the same feature sets (age, blood pressure, BMI etc.) we have no apriori reason to believe that the sets of patients come from the same distribution.

An effective domain adaptation is only possible when common feature representations of source and target domains are found [2]. In the literature of statistics and machine learning, the question of whether two datasets come from the same distribution, also known as the two-sample or homogeneity problem, has been tackled for many years in the context of data integration [3]. However, identifying which features are variant between two datasets is a relatively unexplored problem. Solving this problem is crucial for domain adaptation, since once the invariant features are identified, source and target domains can be reduced to the same distribution, and supervised machine learning algorithms can be applied.

In this paper, we present a novel, reliable and efficient unsupervised method to distinguish variant and invariant features across source and target datasets. We formulate the problem as a convex optimization problem which has obvious advantages such as reliability and efficiency. Experiments on the synthetic and real-world datasets reveal that: 1. Our method can discriminate between variant and invariant features with perfect accuracy 2. Knowing which features are variant is extremely important for domain adaptation: there is 30% improvement in prediction accuracy when only invariant features are used for training as opposed to all features 3. Our method outperforms other state-of-the-art transfer learning approaches on benchmark real-world datasets. Finally, rather than projecting source and target datasets to the feature space, or re-weighting instances to reduce their distance, we introduce a novel transfer learning approach.

The rest of the paper is organized as follows: In the following section, we formalize the problem statement and our solution. In section 3, we describe the experiments we conducted on synthetic and real world datasets. In section 4, we review related work. In the last section, we conclude the paper and provide future prospects.

2 Feature Selection with MMD (f-MMD)

Current methods in information theory can provide a measure of distance between two domains. For example, Kullback-Leibler divergence is a widely used metric to measure distance between two distributions. However, it requires expensive distribution density calculation. As a non-parametric distance measure, Borgwardt et al. proposed Maximum Mean Discrepancy statistic [3]. The main idea is that, under a sufficiently rich reproducing kernel Hilbert space (RKHS), if feature means of the population are identical then it is guaranteed that the distributions are the same [6]. Based on this theorem, distance between samples of two distributions can be measured by the difference of the empirical means of the samples in a RKHS [10]. Formally:

Definition: Let X = $\{x_1, x_2, .., x_m\}$ and Y= $\{y_1, y_2, .., y_n\}$ be two sets of observations drawn from Borel probability distributions p and q. Let $\mathcal{F}$ be a class of functions f: $\mathcal{X} \rightarrow \mathbb{R}$ then the empirical estimate of MMD is defined as :

$$MMD[\mathcal{F}, X, Y] = sup_{f \in \mathcal{F}} \left(\frac{1}{m} \sum_{i=1}^{m} f(x_i) - \frac{1}{n} \sum_{i=1}^{n} f(y_i) \right) \tag{1}$$

As shown by Borgwardt et al. [3] when $\mathcal{F}$ is rich enough, MMD ($\mathcal{F}$, X, Y) will be zero if and only if p = q. However, if it is too rich, for most finite samples of X and Y, MMD will differ significantly from 0. It has been shown that the unit ball in a universal RKHS is a sufficiently large function class that can be chosen as $\mathcal{F}$. Let $\phi(x)$ be the feature map defined as $\phi(x)$: $\mathcal{X} \rightarrow \mathcal{H}$, where $\mathcal{H}$ is a universal RKHS. Function evaluation can then be written as f(x) = $\langle \phi(x), f \rangle$. Based on this argument, the distance between two domains, S and T, can be measured by the squared difference in the empirical means:

$$Dist(X_S, X_T) = \left\| \frac{1}{n_S} \sum_{i=1}^{n_S} \phi(X_{S_i}) - \frac{1}{n_T} \sum_{i=1}^{n_T} \phi(X_{T_i}) \right\|_{\mathcal{H}}^2 \tag{2}$$

Let x and y denote instances from source and target domains respectively: $x \in X_S, y \in X_T$. Following Pan et al., equation (2) can be written in the following form using the kernel trick, i.e. k$(z_i, z_j^T) = \phi(z_i)\phi(z_j^T)$, where k is a positive definite kernel [9]:

$$Dist(X_S, X_T) = tr(KL) \tag{3}$$

such that:

$$K = \begin{bmatrix} K_{xx} & K_{xy} \\ K_{yx} & K_{yy} \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix} \tag{4}$$

where K $\in \mathbb{R}^{(n_S+n_T)\times(n_S+n_T)}$ is a composite kernel matrix and K_{xx}, K_{yy} and K_{xy} are kernel matrices defined by k on the source domain, target domain and cross domains respectively. n_S and n_T denotes the number of instances in the source and target domains. L $\succeq 0$ is a coefficient matrix with $L_{xx} = \frac{1}{n_S^2}$, $L_{yy} = \frac{1}{n_T^2}$ and $L_{xy} = \frac{-1}{n_S * n_T}$.

Our goal is to find the features whose distributions are variant between the two domains. In other words, we are trying to find features which contribute to the distance between two domains the most. Let's define a diagonal weight matrix W, whose diagonal entries correspond to feature weights. Let W $\in \mathbb{R}^{d\times d}$ be this diagonal weight matrix and $x \in \mathbb{R}^{d\times 1}$ be a d dimensional sample vector. Let σ be a feature map such that:

$$\sigma(x) \rightarrow x\, W^{1/2}.$$

Finally, we can define the new (positive definite) kernel, $k'(x_i, x_j^T)$ as:

$$k'(x_i, x_j^T) = \langle \phi \circ \sigma(x_i), \phi \circ \sigma(x_j^T) \rangle_{\mathcal{H}}$$

where $\mathcal{H}$ is a universal RKHS.

The new kernel matrices K'_{xx}, K'_{yy}, K'_{xy} can be defined by k' on the source domain, target domain and cross domain respectively. The new composite kernel matrix K' can be computed with equation (4) on K'_{xx}, K'_{yy}, K'_{xy} (note that K'_{yx} = $(K^{'}_{xy})^T$).

To illustrate the argument with an example, let $\phi(x)$: X → $\mathcal{H}$ be a polynomial kernel with degree, d. Then, K'_{xx} is:

$$K'_{xx} = ((xW^{1/2}) * (xW^{1/2})^T \quad +1)^d$$
$$K'_{xx} = (xWx + 1)^d$$

K'_{yy}, K'_{xy} can be computed in a similar fashion.

To solve for the matrix W, we present the following convex optimization problem:

$$\begin{aligned} W^* = \arg\min_{W} \quad & -trace(K'L) \\ \text{subject to} \quad & diag(W)^T * diag(W) \leq 1 \\ & W > 0 \end{aligned} \tag{5}$$

where diag(W) ∈ $\mathbb{R}^{d\times 1}$ is the diagonal of the weight matrix. Intuition behind this optimization problem is the following:

- By assigning higher weights to features which minimize the negative MMD score in the objective function, we create a gap between the weights of the variant and invariant features across domains. Our assumption here is that there is at least one feature that differs across domains, i.e. the domains are not identical.
- With the first constraint, we are constraining the size of weights by applying a ridge penalty.

Equation (5) is a quadratically constrained quadratic program (QCQP) which can be cast as a Semidefinite Program (SDP). When interior point methods are used, QCQP can have polynomial worst-case complexity [14]. After solving equation (5), using a QCQP solver (in our experiments we used CVX [5][4]), variant features can be found by applying a threshold function to the diagonal entries of W^*. Denoting the set of variant features as V, and the set of invariant features as N, Algorithm 1 describes how we populate V and N. For the rest of this paper we will refer our approach as f-MMD.

3 Experiments

We tested the performance of our algorithm first on the synthetic datasets we designed, and then on the real world datasets. With the synthetic datasets, our purpose is to see how well our algorithm can distinguish between variant and

Algorithm 1. Feature Separation with MMD (f-MMD)

```
Input: Samples from source domain, $X_S$, and target domain, $X_T$, weight threshold $\lambda$
Output: Variant feature set V, and invariant feature set N.
Solve the optimization problem (5) to obtain the weight matrix $W^*$.
w ← diag($W^*$)
for $i = 1 : d$ where d is the number of features do
    if $w_i \geq \lambda$ then
        V ← V ∪ i
    else $\{w_i < \lambda\}$
        N ← N ∪ i
    end if
end for
```

invariant features, and how this information can aid the classification performance. With the experiments on the real world datasets, since we don't know apriori which features are identically distributed (or differently distributed), we only measured the improvement in the prediction performance, after applying our algorithm to select invariant features.

3.1 Synthetic Datasets

Synthetic datasets are designed to address the following questions:

- Can f-MMD identify features whose distribution vary between domains?
- How does the removal of the variant features affect prediction performance?

Synthetic data is generated as follows: Given m, the number of samples from each domain, d, the number of dimensions, k, the number of variant dimensions and t, the number of dimensions related to the class label, we sample invariant (d-k) features from (d-k) randomly picked distributions with zero mean and unit variance. For the first domain, k variant dimensions are sampled from randomly picked k distributions with zero mean and unit variance. For the second domain, these dimensions are sampled from the same k distributions but with linear shift in sample mean. Similar to the random signal generation used in [1], there are 18 distributions from which a feature is sampled: exponential, student (degrees of freedom = 3 or 5), Laplace, mixture of 2 double exponentials, symmetric 2 gauss (multimodal, transmodal, unimodal), uniform, asymmetric 2 gauss (multimodal, transmodal, or unimodal), asymmetric 4 gauss (multimodal, transmodal, or unimodal) and symmetric 4 gauss (multimodal, transmodal, or unimodal). To create class labels, d dimensional weight vector, $v \in R^d$ is drawn from the standard uniform distribution. t features that are related to the class label are randomly selected from all d features. A d dimensional indicator vector is constructed, where $I_i = 1$ if i^{th} feature is related to the class label, 0 otherwise. Consequently, $\sum_{j=1}^{d} I_j = t$. For i = 1...d, we set $v_i = 0$ if $I_i = 0$, otherwise, we left it unchanged. Class labels are then found by applying sign function to the data sample x, i.e. y = sign(v*x).

The illustration of the synthetic data in 2 dimensions are shown in Figure 1. Source data is shown in red, target data is shown in blue. x_1 is the invariant dimension (identically distributed in both domains), x_2 is the variant dimension - there is a linear shift in the means across domains. x_2 is also the predictive dimension for class labels in this example.

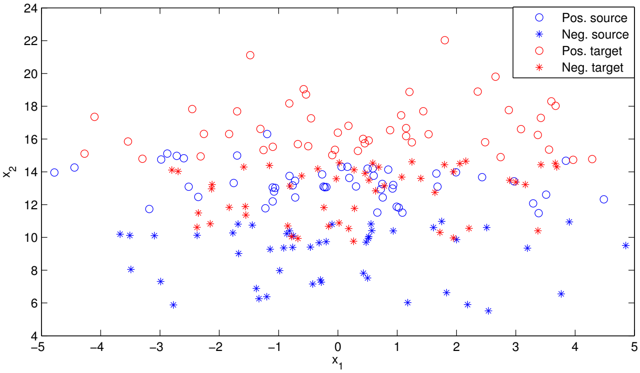


Fig. 1. 2D synthetic data, source data is shown in red, target data is shown in blue. Positive samples are shown with stars, and negative samples are shown with dots.

Next, we created d = 100 dimensional synthetic dataset with 200 samples (100 instances from each domain) by using the procedure described above. Each subfigure in Figure 2 shows the output weights for each dimension enumerated from 1 to 100. As evident from each subfigure, the features that are given significantly higher weights by our algorithm, are indeed variant features.

To illustrate the weight distribution between variant and invariant features, we generated another synthetic dataset with 10 dimensions. This time we varied k from 1 to 10, i.e. we generated 10 source and target datasets with 1 to 10 variant dimensions. The output weights sorted in descending order are given in Table 1. λ parameter for the threshold function can be found empirically by observing the output weight distribution, and picking a value from the range with largest difference in weights. In our experiments, we used $\lambda = 0.1$. Note that the number of features that have weights above 0.1 is exactly the same as k, our algorithm successfully identifies all of the variant features between datasets.

To address the second question, whether prediction performance improves after the removal of variant features, we trained linear SVM and logistic regression on source domain to predict class labels in the target domain. Linear classifiers are chosen to capture the contribution of each feature independently to the classification prediction. We used a source (training) dataset of size 300, with 20 dimensions (10 variant, 10 invariant). We randomly picked 10 features to be related to the class variable, y.

Prediction performances of linear SVM and logistic regression on the synthetic dataset are shown in Table 2 and in Table 3 respectively. First column indicates the prediction accuracy when the classifier is trained only with the invariant

Table 1. Feature weights for each synthetic dataset sorted in descending order. k is the number of variant dimensions in each dataset. The weights for the invariant features are significantly lower than the variant features as expected. Dimensions that have weights above 0.1 are indeed the correct variant dimensions.

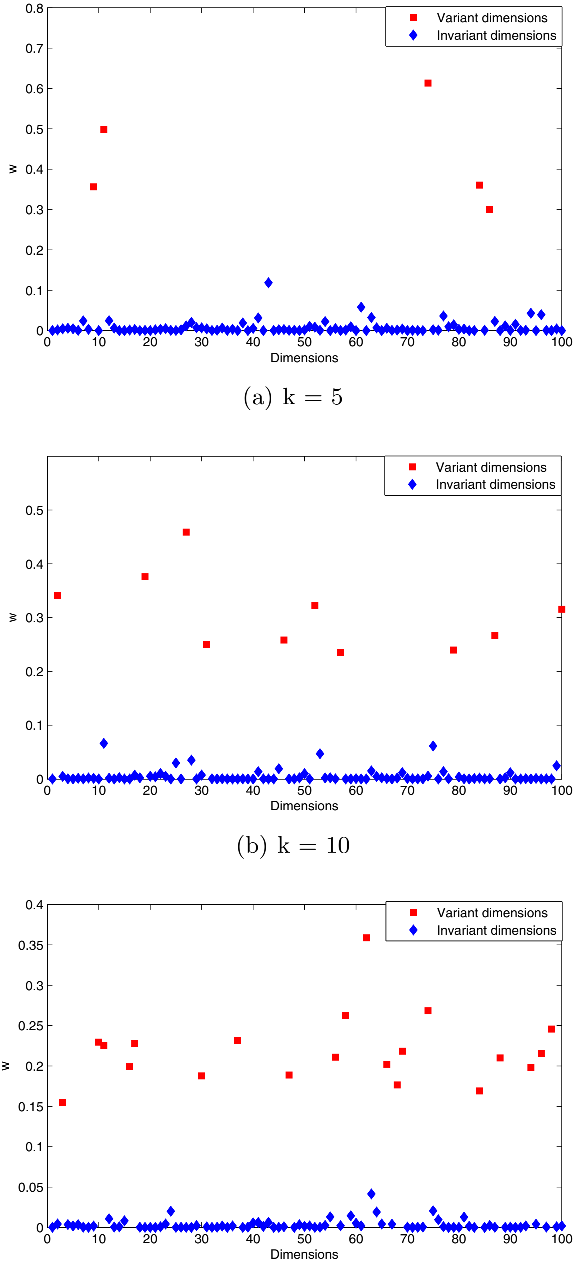
Dimension	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
w_1	0.999	0.827	0.793	0.620	0.531	0.503	0.466	0.621	0.412	0.484
w_2	0.028	0.563	0.470	0.597	0.498	0.464	0.435	0.339	0.406	0.335
w_3	0.027	0.008	0.386	0.420	0.462	0.420	0.414	0.325	0.402	0.311
w_4	0.024	0.007	0.006	0.287	0.440	0.390	0.399	0.292	0.341	0.306
w_5	0.004	0.005	0.003	0.034	0.251	0.328	0.372	0.291	0.315	0.305
w_6	0.003	0.004	0.001	0.000	0.002	0.310	0.285	0.287	0.309	0.295
w_7	0.003	0.002	0.001	0.000	0.001	0.002	0.206	0.267	0.289	0.293
w_8	0.001	0.002	0.000	0.000	0.001	0.001	0.002	0.262	0.289	0.283
w_9	0.000	0.001	0.000	0.000	0.000	0.001	0.000	0.046	0.163	0.266
w_{10}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.001	0.213

features whose f-MMD weights smaller than $\lambda = 0.1$. Second column indicates the accuracy when its trained with all the features. Training sample size is fixed at 300 instances, while the target dataset size is varied from 100 to 300 samples. As can be seen from Table 2 and Table 3, there is a drastic improvement in prediction accuracy when only invariant features are used in training and testing, as opposed to using all the features. This further supports our intuition that using only the invariant features can have significant benefits in domain adaptation.

We also compared dimensionality reduction with f-MMD to other benchmark methods: Transfer Component Analysis (TCA) [10] and kernel PCA (KPCA). TCA is dimensionality reduction method that uses MMD as a distance measure across domains. It learns transfer components that reduce the distance across domains in a RKHS and performs mapping onto the learned transfer components. The prediction performance of the 3 algorithms on the synthetic dataset is shown in Table 4. Total number of features is 40. Source domain size is fixed at 300, and target domain size is increased from 100 to 300. The reduced number of dimensions is the same for all three methods, (d = 20). As can be seen in Table 4, our method outperforms both TCA and kernel PCA, especially as the target domain size increases.

3.2 Real World Datasets

We tested our approach on two real world datasets: USPS handwritten digit images dataset and WIFI localization dataset [16]. Both datasets are commonly used in transfer learning tasks. USPS dataset contains images of size 16 x 16, totaling up 256 features, with pixel values ranging from 0 to 2. Many previous work states that, for the binary classification tasks on the USPS dataset, discriminating 4 from 7 [15] and 4 from 9 is particularly challenging [12][17].



(a) k = 5

(b) k = 10

(c) k = 20

Fig. 2. Weights, with respect to dimensions. Red dots illustrate variant features, blue dots illustrate invariant features. As can be seen in each subfigure, f-MMD weights for variant features are significantly higher than the invariant features.

Hence to make our task harder, for source domain we used digits 4 and 7, and for target domain we used digits 4 and 9. Labeling digit 4 as positive class and 7 and 9 as negative classes, the goal is to transfer the discriminative knowledge between 4 and 7 in the source domain to 4 and 9 in the target domain. Dimension size is determined by the output of f-MMD: Using a threshold, $\lambda = 0.1$, we removed all features with weights larger than λ. Denoting the number of remaining features with p, for f-MMD, linear SVM is trained with the reduced dataset of p-dimensions. For TCA, raw data is first mapped to the feature space, and dimensions with the highest p eigenvalues are used for classification.

WIFI localization dataset consists of wifi data collected in an indoor building at time points A and B. The goal is to predict the location where the wifi data is received at time point B, using the labeled data collected at time point A. This dataset has been used as a benchmark dataset for transfer learning applications, since source domain (data from time point A) and target domain (data from time point B) do not come from the same distribution [16]. For this dataset, we trained a ridge regression with ridge penalty 0.05 to predict the locations. Following [10] we used a training dataset of 621 instances, and varied the testing dataset size from 100 to 500. We compared our approach to KPCA and TCA, keeping the dimension size equal for all 3 methods.

Figure 3 (a) shows average absolute regression error results after applying TCA, KPCA and f-MMD respectively, along with the ridge regression results with no dimension reduction on the WIFI dataset. Figure 3 (b) shows the accuracy of linear SVM after applying TCA, KPCA and f-MMD on the USPS dataset. The exact performance results on WIFI localization dataset and USPS dataset are shown in Table 5 and Table 6 respectively. On both datasets, experiments show that our method significantly outperforms other feature representation methods: On the USPS handwritten digits dataset, for 550 samples, we obtain a 84% classification accuracy, while TCA and Kernel PCA achieves a mere 44% and 78.8% respectively. On the WIFI localization dataset, with the same number of dimensions and with 921 samples, average ridge regression error after dimensionality reduction with our algorithm is 88.6, while after TCA and Kernel PCA it is 103.37 and 92.4 respectively. This shows that our algorithm is a very promising method for domain adaptation.

Table 2. Prediction accuracy of SVM on synthetic dataset

#Samples	Invariant Features	All Features
400	86%	61%
450	86.7%	55.3%
500	86%	55.5%
550	87.2%	57.2%
600	87%	56.7%

Table 3. Prediction accuracy of logistic regression on synthetic dataset

#Samples	Invariant Features	All Features
400	82%	62%
450	84.7%	58%
500	83.5%	59.5%
550	85.2%	60.9%
600	84%	61.3%

Table 4. Linear SVM classification performance after f-MMD, TCA and KPCA on synthetic dataset

#Samples	f-MMD	TCA	KPCA
400	86%	81%	55%
500	86%	75%	48.5%
550	87.2%	67.2%	50.4%
600	87%	58%	51%

Table 5. Average absolute ridge regression error after f-MMD, TCA and KPCA on WIFI localization dataset

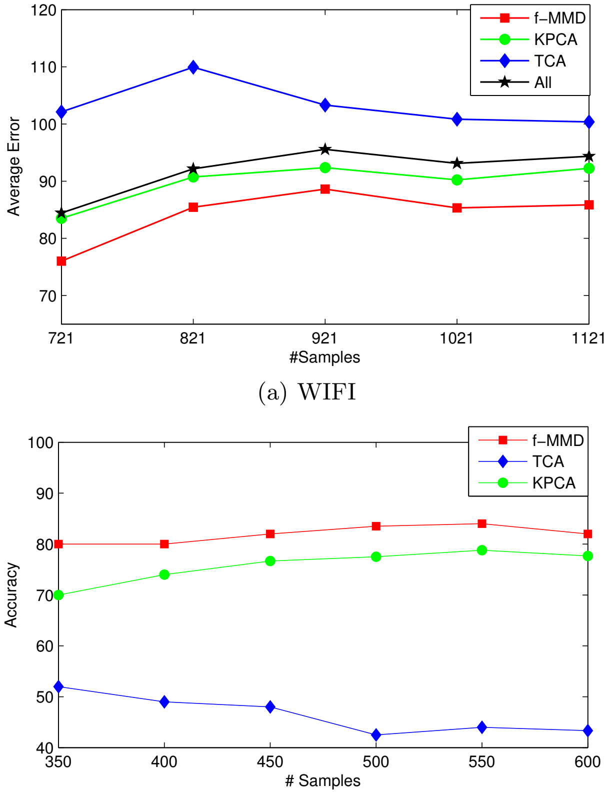
#Samples	f-MMD	TCA	KPCA
721	76.02	102.13	83.51
821	85.44	109.93	90.75
921	88.62	103.32	92.38
1021	85.32	100.84	90.23
1121	85.87	100.37	92.24

Table 6. Classification accuracy of linear SVM after f-MMD, TCA and KPCA on the USPS dataset

#Samples	f-MMD	TCA	KPCA
350	80	52	70
400	80	49	74
450	82	48	76.5
500	83.5	42	77.5
550	84	44	78.8
600	82	43	77.6

4 Related Work

Prior work in domain adaptation focuses on reducing the distance between source and target domains, either through re-weighting the instances (instance based approaches) or finding a common feature representation between two domains (feature based approaches). Instance based approaches, first estimate the weights



(a) WIFI

(b) USPS

Fig. 3. Comparison of f-MMD to benchmark methods. Figure 3 (a) shows average absolute ridge regression error of f-MMD, TCA and KPCA with respect to the sample size, as testing sample size is increased from 100 to 500. Results when no dimensionality reduction is applied is shown with black. Figure 3 (b) shows linear SVM prediction accuracy after f-MMD and TCA with respect to sample size as testing sample size is increased from 50 to 300.

corresponding to each instance in the source domain, prior model training. Instance weights are typically proportional to the distance between source and target density distributions. As a distance measure, Kullback-Leibler divergence [13], or Maximum Mean Discrepancy can be used [7]. Instance based approaches assume that there is a subset of instances with similar distributions in source and target domains, however such a subset may not exist when there are features that are variantly distributed across the two domains.

This work is mostly related to feature based approaches, which assume that the domains share a subset of features that come from similar distributions, and there are features that are variantly distributed across domains. The goal is to find a common feature representation of source and target domains. Among prior work in feature based approaches, Pan et al. first proposed Maximum Mean Discrepancy

Embedding (MMDE) [9] where the distance between distributions are measured with Maximum Mean Discrepancy. In MMDE, first, the kernel that minimizes distance between two distributions is found, and then kernel PCA is applied to the learned kernel. However this method requires an expensive SDP computation, and it is not feasible to be used on large datasets. Hence subsequently Pan et al. proposed Transfer Component Analysis (TCA), where the goal is to find a projection that minimizes distance between distributions [10]. TCA doesn't require an SDP computation, it is shown to be more efficient than MMDE in domain adaptation problems such as WIFI localization prediction [10].

5 Conclusion

In this paper, we proposed a novel and an extremely efficient method for domain adaptation. Unlike previous feature based approaches, rather than finding a projection of the feature space to maximize the similarity between source and target domains, we identify the features whose distribution vary between the two domains. In our experiments, we showed that knowing which features are variant and incorporating this knowledge to the prediction task significantly improves prediction performance, a novel finding in domain adaptation. We showed that our method significantly outperforms other comparable feature based methods on benchmark datasets. In the future, our goal is to extend this work to select the kernels that are differently distributed across domains. We are also intrigued to see how we can incorporate variant features to further increase prediction accuracy in domain adaptation.

References

1. Bach, F.R., Jordan, M.: Kernel Independent Component Analysis. Journal of Machine Learning Research 3, 1–48 (2002)
2. Blitzer J., McDonald R., Pereira F.: Domain Adaptation with Structural Correspondence Learning. In: EMNLP 2006: 2006 Conference on Empirical Methods in Natural Language Processing, pp. 120–128 (2006)
3. Borgwardt, K., Gretton, A., Rasch, M., Kriegel, H.P., Schölkopf, B., Smola, A.J.: Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics 22(14), 49–57 (2006)
4. Grant, M., Boyd, S.: Graph implementations for nonsmooth convex programs. In: Blondel, V., Boyd, S., Kimura, H. (eds.) Recent Advances in Learning and Control (a tribute to M. Vidyasagar). LNCIS, pp. 95–110. Springer, Heidelberg (2008)
5. Grant, M., Boyd, S.: CVX: Matlab software for disciplined convex programming, version 1.21 (April 2011), http://cvxr.com/cvx
6. Gretton, A., Borgwardt, K.M., Rasch, M., Schölkopf, B., Smola, A.: A Kernel Method for the Two-Sample-Problem. In: Advances in Neural Information Processing Systems. Proceedings of the 2006 Conference, vol. 19, pp. 513–520. MIT Press, Cambridge (2007)
7. Huang, J., Smola, A.J., Gretton, A., Borgwardt, K.M., Schölkopf, B.: Correcting sample selection bias by unlabeled data. Advances in Neural Information Processing Systems 19, 601–608 (2007)

8. Margolis, A.: Literature Review of Domain Adaptation with Unlabeled Data
9. Pan, S.J., Kwok, J.T., Yang, Q.: Transfer Learning via Dimensionality Reduction. In: AAAI, pp. 677–682 (2008)
10. Pan, S.J., Tsang, I.W., Kwok J.T., Yang, Q.: Domain Adaptation via Transfer Component Analysis. In: Proceedings of IJCAI 2009, pp. 1187–1192 (2009)
11. Pan, S.J., Yang, Q.: IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2010)
12. Remus, S., Tomasi, C.: Semi-Supervised Fisher Linear Discriminant (SFLD). In: ICASSP, pp. 1862–1865 (2010)
13. Sugiyama, M., Nakajima, S., Kashima, H., Bnau, P.V., Kawanabe, M.: Direct Importance Estimation with Model Selection and Its Application to Covariate Shift Adaptation. In: NIPS (2007)
14. Vandenberghe, L., Boyd, S.: Semidefinite Programming. SIAM Review 38(1), 49–95 (1996)
15. Xu, Z., Jin, R., Lyu, M.R., King, I.: Discriminative Semi-Supervised Feature Selection via Manifold Regularization. In: IJCAI, pp. 1303–1308 (2009)
16. Yang, Q., Pan, S.J., Zheng, V.W.: Estimating Location Using Wi-Fi. IEEE Intelligent Systems 23(1), 8–13 (2008)
17. Zeng, H., Cheung, Y.: Feature Selection for Local Learning Based Clustering. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 414–425. Springer, Heidelberg (2009)

Multiview Semi-supervised Learning for Ranking Multilingual Documents

Nicolas Usunier[1], Massih-Reza Amini[2], and Cyril Goutte[2]

[1] Université Pierre et Marie Curie, LIP6, F-75252 Paris 5 cedex, France
[2] National Research Council Canada, IIT, Gatineau, QC J8X 3X7, Canada

Abstract. We address the problem of learning to rank documents in a multilingual context, when reference ranking information is only partially available. We propose a multiview learning approach to this semi-supervised ranking task, where the translation of a document in a given language is considered as a view of the document. Although both multiview and semi-supervised learning of classifiers have been studied extensively in recent years, their application to the problem of ranking has received much less attention. We describe a semi-supervised multiview ranking algorithm that exploits a global agreement between view-specific ranking functions on a set of unlabeled observations. We show that our proposed algorithm achieves significant improvements over both semi-supervised multiview classification and semi-supervised single-view rankers on a large multilingual collection of Reuters news covering 5 languages. Our experiments also suggest that our approach is most effective when few labeled documents are available and the classes are imbalanced.

Keywords: Learning to Rank, Semi-supervised Learning, Multiview Learning.

1 Introduction

We address the problem of ranking multilingual documents. Ranking is an important problem in several applications related to Information Retrieval such as search, or summarization. Although multilingual document collections are very common in many national or supranational contexts, the bulk of document organization techniques is still developed in a monolingual setting, often for English. We aim at developing ranking tools for handling such multilingual collections in ways smarter than using independent monolingual approaches. We also consider situations where only partial supervision is available in the form of reference ranking information.

In order to learn in a multilingual setting, our proposal relies on the framework of *multiview* learning. In a parallel corpus of multilingual documents, we consider each language as a separate *view* of a document. Each document will therefore have as many views as there are languages in the corpus. Earlier work suggests that this framework is an efficient way to learn classifiers in a multilingual setting [3]. We show how multiview learning can be extended to ranking,

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 443–458, 2011.

and how it can be applied to ranking multilingual documents. More specifically, we are interested in *bipartite* ranking problems such as information routing [23], in which we seek a linear ordering of objects that belong to two relevance judgments, such that relevant examples are ranked higher than irrelevant ones. This task has been extensively studied in the supervised learning setting [9,11,16] due to its practical importance. It is also a first step towards more general ranking tasks, where the reference ranking information can take the form of an arbitrary preference relation over the examples [13]. In addition, a common issue with tasks involving large collections of textual documents is that providing extensive human supervision (such as category labels or ranking information) can be prohibitively expensive. Semi-supervised learning techniques have been developed to address this problem. In the framework of multiview learning for classification, these approaches use the labeled data to train several view-specific predictors, and rely on the intuition that these predictors should have similar predictions on the unlabeled set. This additional constraint may reduce the possible choices of predictors, leading to better generalization guarantees [24]. Our approach to semi-supervised multiview ranking (SmVR) follows the same intuition. Given score functions $(h_1, ..., h_V)$ independently trained on each view, we define a notion of global agreement between them as the expectation, over random pairs of objects (x, x'), that two score functions $(h_v, h_{v'})$ predict the same relative ordering. We hence describe a learning process in which language or view-specific ranking functions should achieve high ranking performance on the labeled training set, while minimizing a disagreement measure between each other on the unlabeled dataset.

We propose an efficient multilingual ranking algorithm inspired by iterative co-training techniques [7]. Our method exploits randomization and efficient algorithms for supervised bipartite ranking to break the quadratic complexity (with respect to the number of unlabeled objects) inherent to the SmVR approach based on the minimization of the disagreement. Experiments carried out on a multilingual text corpus indicate that SmVR provides a significant improvement over both single-view semi-supervised ranking and semi-supervised multiview classification, and is more robust to class imbalance than a state-of-the-art semi-supervised multiview classification algorithm. Promising results have also been published on semi supervised ranking in the single-view setting [2,14,22], but to the best of our knowledge, none was extended to multiview learning.

In the next section, we briefly review some related state-of-the-art. In Section 3, we present our solution to semi-supervised ranking in a multiview setting, and Section 4 describes the algorithm applied in our experiments. The experimental results are reported in section 5.2, where we show that our method is effective on a large multilingual collection of Reuters documents covering five languages.

2 Related Work

In this section, we review the state-of-the-art on bipartite ranking, multiview learning and semi-supervised learning for classification and ranking.

2.1 Bipartite Ranking

The task of learning to rank was introduced by Cohen et al. [10], motivated by information retrieval applications where the results take the form of an ordered list of objects. The new framework introduced an algorithm with the ability to learn from a new form of supervision, namely preference relations over the examples. The algorithm also optimized some criteria related to the ranking performance of the predictor. While that original ranking algorithm learned a preference relation on the example space, subsequent proposals reduced the task to learning a scoring function [15,13]. The ranking is then created by sorting the examples by decreasing scores. Bipartite ranking is the special case of ranking where the supervision is a bipartite graph [13]. It corresponds to information routing problems where the query (or topic) is fixed and examples are either relevant or irrelevant to the query [23]. Bipartite ranking can be formulated as the learning of a scoring function by optimizing the area under the ROC curve (AUC, see Section 3) [1]. While many classification algorithms produce scores and thus can be used in the context of bipartite ranking, Cortes & Mohri [11] analyze the advantage of optimizing the AUC instead of the classification accuracy when one searches good ranking performance. Their conclusion is that ranking methods should be superior when the data is imbalanced (a vast majority of the examples belong to the same class) or noisy. The theory underlying bipartite ranking has been extensively studied [9] and efficient algorithms for AUC optimization have been designed [13,16]. From an algorithmic perspective, the extension of supervised learning algorithms from bipartite ranking to the general case is usually straightforward, even though the computational cost might significantly increase. Most works on bipartite ranking were done in the supervised and single view setting. We propose here an extension to semi-supervised multiview learning.

2.2 Multiview Learning

Multiview learning deals with observations that can be described in several representation spaces, such that each representation space may be used to build a predictor. Multilingual documents can naturally be seen as multiview observations: each language in which a document is translated corresponds to a view. The overall goal of multiview learning is to combine predictors over each view (called *view-specific* predictors) in order to improve the overall performance beyond that of predictors trained on each view separately, or on trivial combinations of views. The first successful multiview learning technique was Blum's co-training algorithm [7] which iteratively labels unlabeled examples based on predictors trained in different views. A related approach is co-regularization [24] where the view-specific predictors are constrained to produce similar predictions. Other notable multiview techniques are multiple kernel learning approach (MKL, e.g. [4]) , and techniques relying on (kernel) Canonical Correlation Analysis [18] or multiview Fisher Discriminant Analysis [12]. Note that although co-training [7] and co-regularization [24] have different theoretical backgrounds and motivation, empirical evidence shows that view-specific classifiers trained by iterative co-training algorithm tend to agree on the pool of unlabeled data. The pseudo-labeling method of co-training can thus be seen as an iterative method for increasing the agreement between predictors. This issue will be at the core of our approach (see Section 4). Although multiview learning has been used from its origin on textual data [7], it has only recently been applied to multilingual data [3]. Moreover, the multiview framework has been extensively studied for classification tasks, but its use in bipartite ranking is novel.

2.3 Semi-supervised Classification

Apart from multiview approaches, the field of single view semi-supervised learning has been an active area of research since the late nineties [27]. The overall aim is to design algorithms which are able to extract information from both labeled and unlabeled data to improve performance. While some work on semi-supervised learning deals with ranking tasks, the main focus was classification. Most studies on the semi-supervised paradigm rely on the *cluster assumption*, which states that examples within a given cluster are likely to be of the same class. Algorithms designed for this assumption are generally based on mixture models [21]. Semi-supervised discriminative approaches are mainly based on a similar but slightly different assumption of *low density separation*, which states that high-density regions do not contain the decision boundary [8]. These approaches are mostly iterative algorithms designed to propagate the class labels in the high density regions. Another marginally different assumption is the *manifold assumption*, which holds when high dimensional data lie on a low-dimensional manifold [6]. In such cases, the learning algorithm can avoid the curse of dimensionality which may affect generative models by operating in a low-dimensional space [5]. While both supervised learning for ranking and semi supervised learning for classification have been widely studied in the past, the combination of semi-supervised learning for ranking has just begun to be explored.

2.4 Single View Semi-supervised Ranking

Both supervised learning and our approach to multiview, semi-supervised learning of ranking functions in the bipartite setting are inspired by algorithms for binary classification. The approaches to single view semi-supervised learning of classifiers, however, cannot be easily adapted to ranking. Indeed, the assumptions used in single view semi-supervised classification are such that the decision boundary is easy to detect on the set of unlabeled data. The task of ranking, however, is not about detecting a decision boundary, but rather a scoring function that induces the best possible complete ordering of the observations. This ranking is given by scoring the observations according to their probability of being relevant [9], an information that is not considered by classification criteria: these algorithms only need the most probable class label for a given observation. Some work has been done on single view semi-supervised bipartite ranking with promising experimental results. In [2], an iterative pseudo-labeling step uses neighborhood information while optimizing a ranking objective function on labeled (and pseudo-labeled) training sets. In [22], the unlabeled data is used to change the representation space of the examples, motivated by cases where the class conditional distributions are gaussian. These methods rely on the fact that bipartite ranking data has the form of binary classified data. It is unclear whether these approaches can be extended to more general ranking formulations. In contrast, our multiview method uses a pseudo-labeling step induced by the ranking on the unlabeled data, which should be easier to extend to more general forms of feedback. To the best of our knowledge, all works on semi-supervised ranking have been done in the single view setting. Through the use of multiple views, our approach naturally takes into account the ranking information on the unlabeled set to improve the rankers' performance.

3 Semi-supervised Multiview Learning for Ranking

We present the framework of multiview, semi-supervised ranking with bipartite feedback. We then describe the learning principle underlying our algorithm, presented in Section 4.

3.1 Framework

In bipartite ranking problems, the labeled data take the form of a set $Z = (\mathbf{x}^i, y^i)_{i=1}^n$ of (observation, target) pairs, where $y^i \in \{-1, +1\}$ is called the relevance of observation $\mathbf{x}^i$. Following the standard assumption in machine learning, we assume these examples to be sampled i.i.d. from some fixed (but unknown) distribution, and we denote by (X, Y) a generic pair of random variables which follows that distribution. In a semi-supervised learning setting, we also assume we have access to a pool of unlabeled examples $U = (x^{n+j})_{j=1}^m$ which are i.i.d. and follow the same distribution as X.

In the single-view setting, the goal of bipartite ranking is to learn a function h which assigns a score to any possible input, so that relevant observations (i.e. those with $y = +1$) obtain higher scores than irrelevant ones. The ranking criterion to be optimized is usually taken as the Area Under the ROC Curve (AUC). As shown in [9], the goal of learning is then to minimize the ranking risk:

$$L(h) = \mathbb{P}\big((Y - Y')sgn(h(X) - h(X')) < 0\big) \tag{1}$$

where (X', Y') is an independent copy of (X, Y), $sgn(t) = 2\mathtt{I}_{\{t \geq 0\}} - 1$ is the sign function and $\mathtt{I}_{\{.\}}$ is the indicator function. This risk can be estimated on the labeled set by a U-statistics (which is the AUC, up to an affine transformation):

$$\hat{L}_Z(h) = \frac{1}{n(n-1)} \sum_{i,j} \mathtt{I}_{\{y_i > y_j\}} \mathtt{I}_{\{h(\mathbf{x}^i) \leq h(\mathbf{x}^j)\}}$$

In multiview learning, an observation (in our case, a multilingual document) $\mathbf{x} = (x_1, ..., x_V)$ is described in several representation spaces $\mathcal{X}_v, v \in \{1 \dots V\}$, such that each representation (here, a translation in a given language) x_v can be used to build a predictor. Following the framework of [25] for multiview classification or regression, we can define the objective of multiview ranking as jointly learning *view-specific* scoring functions $h_v : \mathcal{X}_v \rightarrow \mathbb{R}$ (in our case, h_v only considers the translation of the documents in the v-th language) so that their average risk is small, where the joint learning of these view-specific predictors consists in constraining them to agree with each other (i.e. have similar predictions). Such a principle is amenable to semi-supervised learning since the agreement between predictors can be measured without knowing the labels of the observations, and can thus be estimated (and optimized) from the pool of unlabeled data. Since constraining the view-specific predictors to have a low disagreement reduces the function space, one can then expect better generalization guarantees using semi-supervised multiview learning than using plain supervised learning.

More formally, suppose we are given V view-specific scoring function sets $\mathcal{H}_1, ..., \mathcal{H}_V$ and a *disagreement* function $D : \mathcal{H}_1 \times ... \times \mathcal{H}_V \rightarrow [0, 1]$ (the exact definition of D is given in the next subsection). We can then define:

$$\forall t \in [0,1], \mathcal{H}(t) = \{(h_1, .., h_V) \in \mathcal{H}_1 \times ... \times \mathcal{H}_V : D(h_1, \ldots, h_V) \leq t\} \ , \qquad (2)$$

which is the set of tuples $(h_1, ..., h_V)$ which have a disagreement smaller than $t \in [0, 1]$. Using VC dimension [9] or Rademacher complexity arguments [26] to obtain uniform generalization error bounds for ranking, we can find some function $\mathcal{R}_n(\mathcal{H}(t)\,, \delta)$ which increases with t, such that for any t and $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the random draws of Z, we have:

$$\forall (h_1, ..., h_V) \in \mathcal{H}(t)\,, \frac{1}{V} \sum_{v=1}^{V} L(h_v) \leq \frac{1}{V} \sum_{v=1}^{V} \hat{L}_Z(h_v) + \mathcal{R}_n(\mathcal{H}(t)\,, \delta) \ . \qquad (3)$$

This error bound gives us the principle of semi-supervised, multiview ranking: after an appropriate design of the disagreement function D so that it can be estimated on the pool of unlabeled data, the learning algorithm will aim at optimizing the generalization guarantee Eq. (3) by searching among the view-specific scoring functions with small empirical ranking risk, a tuple $(h_1, ..., h_V)$ with a small empirical disagreement on the pool of unlabeled data.

3.2 Disagreement for Bipartite Ranking

The semi-supervised multiview learning process described above is linked to an appropriate measure of disagreement between view-specific scoring functions. Since the ranking risk (and the AUC) linearly decompose into pairwise comparisons between scores, a natural measure of disagreement between two scoring functions h_v and $h_{v'}$ is the probability, over any two random observations, that they do not predict the same ordering:

$$D(h_v, h_{v'}) = \mathbb{P}\big(sgn(h_v(X) - h_v(X')) \neq sgn(h_{v'}(X) - h_{v'}(X'))\big) \,,$$

which can be estimated on the *unlabeled* data set U by:

$$\widehat{D}_U(h_v,h_{v'}) = \frac{1}{m(m-1)}\sum_{i\neq j} \mathbb{I}_{\left\{sgn\left(h_v(x_v^{n+i})-h_v(x_v^{n+j})\right)\neq sgn\left(h_{v'}(x_v^{n+i})-h_{v'}(x_v^{n+j})\right)\right\}} \, .$$

We may note that the empirical disagreement is exactly Kendall's tau between the two rankings predicted on U by h_v and $h_{v'}$. This notion of disagreement (and its empirical counterpart) can then be extended to more than two views by taking the average disagreement between scoring functions for any pair of views:

$$D(h_1,\ldots,h_V){=}\frac{2}{V(V{-}1)}\sum_{v<v'}D(h_v,h_{v'}) \text{ and } \widehat{D}_U(h_1,\ldots,h_V){=}\frac{2}{V(V{-}1)}\sum_{v<v'}\widehat{D}_U(h_v,h_{v'}) \, . \tag{4}$$

Continuing the generalization error bound of Eq. (3), we can note that the empirical disagreement also has the form of a U-statistics, so that VC-dimension or Rademacher arguments can also be used to obtain a uniform (over the whole set of functions $\mathcal{H}=\mathcal{H}_1\times\ldots\times\mathcal{H}_V$) bound on $D(h_1,\ldots,h_V)-\widehat{D}_U(h_1,\ldots,h_V)$. Denoting $\mathcal{G}_m(\mathcal{H},\delta)$ such a bound, we have:

$$\mathbb{P}\Big(\sup_{h_v\in\mathcal{H}_v}\left[D(h_1,\ldots,h_V)-\widehat{D}_U(h_1,\ldots,h_V)\right]\leq\mathcal{G}_m(\mathcal{H},\delta)\Big)\geq 1-\delta \, ,$$

where the probability is taken over U. Using the union bound and plugging this bound into Eq. (3), we have, for any $t\in[0,1]$, with probability at least $1-2\delta$ over both Z and U:

$$\forall(h_1,...,h_V) \text{ s.t. } \widehat{D}_U(h_1,\ldots,h_V)\leq t,$$
$$\frac{1}{V}\sum_{v=1}^{V}L(h_v)\leq\frac{1}{V}\sum_{v=1}^{V}\hat{L}_Z(h_v)+\mathcal{R}_n(\mathcal{H}(t^*),\delta)\, , \text{ with } t^*=t+\mathcal{G}_m(\mathcal{H},\delta) \, .$$

When the unlabeled dataset is large (which is typically the case in semi-supervised learning), $\mathcal{G}_m(\mathcal{H},\delta)$ will be small so that the empirical disagreement will be close to the true one. Thus, considering the last error bound, one can see that when there are many empirical risk minimizers in $\mathcal{H}$ (which is typically true when the labeled training set is very small), we may expect much better generalization guarantees for tuples $(h_1,...,h_V)$ with low disagreement. This is precisely what the algorithm presented in the next section aims at, by iteratively finding view-specific scoring functions with decreasing disagreement (and small empirical risk) using a co-training like procedure, until the disagreement does not improve.

Remark 1. *The authors of [25] argue that the notion of disagreement used in multiview learning should be closely related to the definition of risk, in the sense that they should satisfy a so-called* inverse Lipschitz condition *(see Assumption 2 of [25]). In our case of bipartite ranking, a Bayes-optimal predictor is $\rho(x)=\mathbb{P}(Y=1|\,X=x)$ [9], and, using our notion of disagreement, the excess risk of any* *scoring function h can be written as $L(f) - L(\rho) = \mathbb{E}\big[|\rho(X) - \rho(X')|\, D(h, \rho)\big]$. With a low-noise assumption for ranking similar to the one used by [9] (formally: $\exists c > 0, \exists \alpha \in (0, 1)$ such that $\mathbb{E}\big[|\rho(X) - \rho(X')|^{-\alpha}\big] \leq c$), we can show that $D(h, \rho) \leq \sqrt{c}\,(L(f) - L(\rho))^{\alpha/2}$, which is precisely an inverse Lipschtz condition of [25]. Thus, in low-noise settings for bipartite ranking, one can obtain strong theoretical results with our notion of disagreement, similar to those of Theorem 2 of [25] (up to a straightforward extension of their framework to ranking).*

4 Algorithm

The learning process described above states that we should look for view-specific functions with high `AUC` on the labeled training set, while minimizing the disagreement between the view-specific rankers on the unlabeled dataset.

To that end, we propose an algorithm inspired by pseudo-labeling techniques like iterative co-training [7]. Our approach relies on a supervised learning algorithm for bipartite ranking, and iteratively trains independent rankers on each view with a pseudo-labeling technique: at each round, some unlabeled examples are added to the training set, and their target value is set using the consensus prediction of the view-specific rankers of the previous iteration.

In classification tasks, the pseudo-labeling consists of aggregating the class labels predicted by the view-specific classifiers, for instance taking a majority vote. The unlabeled examples added to the training set at each round are chosen using a measure of confidence in the pseudo-label, in order to avoid adding incorrectly labeled examples to the training material. Although the pseudo-labeling technique used in iterative co-training is not intended to minimize the disagreement between different views, it does empirically tend to decrease the disagreement on the unlabeled set because each classifier is trained with an increasing portion of examples pseudo-labeled by the other classifier. Pseudo-labeling techniques are thus a natural heuristic for learning functions with low disagreement.

Considering our notion of empirical disagreement Eq. (4), it is then natural to define a notion of pseudo-labeling on *pairs* of unlabeled observations: a pair $(\mathbf{x}^{n+i}, \mathbf{x}^{n+j})$ would be labeled $+1$ if the various view-classifiers agree on $h_v(x_v^{n+i}) > h_v(x_v^{n+j})$, and -1 if they agree on the inverse relative ordering. After pseudo-labeling, we would then obtain a training set with pseudo-pairwise preferences (instead of pseudo labels in $\{-1, 1\}$). From a computational point of view, however, this procedure would be extremely costly for two reasons. First, it would require a pass over all pairs of unlabeled inputs at each round. Since there are about m^2 pairs, this is too large by an order of magnitude. Secondly, the pairs of unlabeled inputs selected to be added in the training set do not have the structure of a proper bipartite ranking. The underlying supervised learning algorithm should then be an algorithm that can deal with arbitrary pairwise preferences, which have $\Omega(\ell^2)$ space and time complexity (ℓ is the number of objects in the training set). By contrast, efficient algorithms for bipartite ranking like RankBoost [13] or `SVM`^{multi} [16] run in time $\tilde{O}(\ell)$ and require $O(\ell)$ space.

Algorithm 1. Semi-supervised Multiview Ranking

```
Input:
▷ supervised bipartite ranking algorithm: $\mathcal{A}$;
▷ size of the random pairs sample: $S$;
▷ labeled $Z = (\mathbf{x}^i, y^i)_{i=1}^n$, and unlabeled $U = (\mathbf{x}^{n+j})_{j=1}^m$ multiview training data;
Initialize:
for each view, train $h_v^{(0)}$ on $Z$ with $\mathcal{A}$.
$t \leftarrow 0$;
repeat
    for $s = 1..S$ do
        $(i, j) = \text{sample}\big(\{(k, \ell) \in \{1, ..., m\}^2, k \neq \ell\}\big)$
        if $\forall v, h_v^{(t)}(x_v^{n+i}) > h_v^{(t)}(x_v^{n+j})$ then
            $Z \leftarrow Z \cup \big\{(x^{n+i}, +1), (x^{n+j}, -1)\big\}$
        else if $\forall v, h_v^{(t)}(x_v^{n+i}) < h_v^{(t)}(x_v^{n+j})$ then
            $Z \leftarrow Z \cup \big\{(x^{n+i}, -1), (x^{n+j}, +1)\big\}$
        end if
    end for
    $t \leftarrow t + 1$;
    for each view, train $h_v^{(t)}$ on $Z$ with $\mathcal{A}$;
until $\hat{D}_U\big(h_1^{(t)}, ..., h_V^{(t)}\big) \geq \hat{D}_U\big(h_1^{(t-1)}, ..., h_V^{(t-1)}\big)$
Output: $\forall v \in \{1, .., V\}, h_v^{(t)}$;
```

4.1 Weighted Pseudo-labeling

Our multiview approach to semi-supervised bipartite ranking follows existing iterative pseudo-labeling methods for classification, but relies on two ingredients to reduce the overall time and space complexity to $\tilde{O}(n + m)$.

The first one is a reduction from the pseudo-labeled pairs to bipartite ranking in order to use efficient learning to rank algorithms. The second one is a straightforward random sampling of pairs at each iteration rather than considering all possible pairs of unlabeled examples.

The algorithm is fully described in Algorithm 1. In an initialization step, each view-specific ranker is trained independently on the labeled training set. Then, the algorithm iteratively re-trains one ranker per view on increasing training sets composed of the initial labeled examples, and additional pseudo-labeled examples. The first step of each iteration is the pseudo-labeling step, where we increase the size of the labeled training set. Following iterative pseudo-labeling methods for classification (but applied here to pairs of inputs) we pass through unlabeled pairs and decide whether or not they contain information that should be added to the training set based on a measure of confidence in the pseudo-label.

Following [3], we select only the pairs of examples for which all the view-specific rankers agree on the relative ordering. This requirement of unanimity may be too restrictive when there are many views, but we observed that it works very well in practice (see section 5.2). It is not a major point of our algorithm and can be relaxed if too many pairs are ignored in this step.

Once the pairs are selected, we do not add them directly in the training set for computational reasons: bipartite ranking algorithms are very efficient because their implementation makes heavy use of bipartite structure of the preference graph. In order to keep the preference graph bipartite, we actually add binary labeled inputs: for each selected example pair, the input which is scored higher by all view-specific rankers is added to the training set with label +1, and the other example in the pair is added with label −1. The crucial point here is that examples may be added several times, possibly with differing labels. Therefore, examples are pseudo-labeled and implicitly weighted in the training set, so that the algorithm will learn to rank the unlabeled examples according to how many pairs they appear in as the greater or lower element.

If this process was applied to the entire unlabeled set, it would require a pass over all pairs of unlabeled inputs, leading to $O(m^2)$ time complexity. In order to avoid this overwhelming cost, we randomly select a much smaller number of pairs (in our experiments, we sample $15,000$ pairs at each iteration, from a set of $60,000$ unlabeled examples).

The iterative procedure is repeated until the disagreement does not decrease after re-training. In order to avoid the costly computation of the disagreement at each iteration, it is estimated using the pairs sampled at the current iteration.

4.2 Supervised Ranking Algorithm

Our semi-supervised process relies on an underlying efficient algorithm for learning bipartite ranking functions in a fully supervised setting. In this paper, we use a linear `SVM` for ranking, since linear functions with a bag-of-words representation are known to perform very well on textual data.

For each view, v, denoting $Z = (\mathbf{x}^i, y^i)_{i=1}^{\ell}$ the training set available at some given iteration of the algorithm, we learn a linear scoring function h_v of the form $h_v(x) = \langle \mathbf{w}_v, x_v \rangle$ where $\langle ., . \rangle$ is the dot product in Euclidian space, x_v denotes the bag-of-words representation of document $\mathbf{x}$ in the v-th language, and $\mathbf{w}_v$ is the parameter vector to be learnt for view v.

Learning is carried out by minimizing the following pairwise loss for each view (see e.g. [16]):

$$\mathbf{w}_v = \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + \sum_{i:y^i=1} \sum_{j:y^j=-1} \max\big(0, 1 - \langle \mathbf{w}, x_v^i - x_v^j \rangle\big)$$

The optimization is carried out with an algorithm similar to SVM^{multi} [16], which has time and space complexity in $\tilde{0}(\ell)$. Note that the training set we consider has only binary-labeled pseudo-examples, but some may appear several times with the same or a different pseudo-label as described before.

5 Experimental Results

We illustrate and validate the usefulness of our algorithm on a large collection of documents covering five languages and six categories. We also investigate

Table 1. Number of documents per language (left) and per class (right) in the Reuters RCV1/RCV2 sub-collection considered in our experiments.

	# docs	# docs	Total
Language	source	translated	docs
English	18,758	92,982	111,740
French	26,648	85,092	111,740
German	29,953	81,787	111,740
Italian	24,039	87,701	111,740
Spanish	12,342	99,398	111,740

Topic	# docs	(%)
C15	18,816	16.84
CCAT	21,426	19.17
E21	13,701	12.26
ECAT	19,198	17.18
GCAT	19,178	17.16
M11	19,421	17.39

the impact of the number of annotated documents and the imbalance between relevant and irrelevant documents.

We use a publicly available[1] multilingual multiview text categorization corpus extracted from the Reuters RCV1/RCV2 corpus [3], summarized in Table 1. The corpus is originally *comparable* but was made into a parallel, multiview corpus by translating each original document in all other languages. Each of the 111,740 documents is available in 5 views: original language and four translations. The second column in Table 1 indicates the distribution of source languages for our collection. All documents (originals and translations) were indexed using a standard preprocessing chain and are available already indexed.

For each topic, the bipartite ranking problem is to rank documents within this topic above documents belonging to the other topics. The evaluation is carried out with two standard Information Retrieval metrics: the Average Precision (AvP) and Area Under the ROC Curve (AUC) [20]. Each experiment is performed over 10 random splits (labeled training/unlabeled training/test) of the initial collection. The test split always contains 25% of the documents. All labeled/unlabeled/test splits respect the initial topic and language proportions.[2]

5.1 Models

In order to evaluate the benefits of the semi-supervised, multiview approach, and justify our focus on bipartite ranking as opposed to classification, we compare the following five models:

sVR-SVM: fully supervised, single-view ranking. Train monolingual ranking functions on each view, on labeled data only. It corresponds to $h_v^{(0)}$ in Algorithm 1, uses no unlabeled data, and trains independent monolingual rankers.

SsVR-SVM: semi-supervised single-view ranking. Iterative pseudo-labeling approach propagating labels to neighbouring unlabeled examples, as in [2] but using a SVM ranker instead of boosting.

Conc-SR: semi-supervised single-view ranking on concatenated views. Same as the previous model, but operating on concatenated views, instead of independently on each view.

[1] http://multilingreuters.iit.nrc.ca/

[2] With a minimum of 2 positive examples in each labeled training set.

Table 2. AUC and AvP for four competing models, starting from 10 labeled training examples, averaged over 10 random splits and five languages, keeping real class proportions. $^{\downarrow}$ indicates the performance is significantly worse than the best result (in bold). SmVR-SVM is Algorithm 1.

Strategy	C15		CCAT		E21		ECAT		GCAT		M11	
	AUC	AvP	AUC	AvP	AUC	AvP	AUC	AvP	AUC	AvP	AUC	AvP
sVR-SVM	.669$^{\downarrow}$	.329$^{\downarrow}$	.624$^{\downarrow}$	.291$^{\downarrow}$	.621$^{\downarrow}$	.265$^{\downarrow}$	.638$^{\downarrow}$	.283$^{\downarrow}$	.755$^{\downarrow}$	.418$^{\downarrow}$	.811$^{\downarrow}$	.566$^{\downarrow}$
SmVC-SVM	.698$^{\downarrow}$	.334$^{\downarrow}$	.645$^{\downarrow}$	.312$^{\downarrow}$	.652$^{\downarrow}$	.282$^{\downarrow}$	.649$^{\downarrow}$	.294$^{\downarrow}$	.773$^{\downarrow}$	.434$^{\downarrow}$	.821$^{\downarrow}$	.591$^{\downarrow}$
SsVR-SVM	.724$^{\downarrow}$	.416$^{\downarrow}$	.658$^{\downarrow}$	.324$^{\downarrow}$	.665$^{\downarrow}$	.306	.662$^{\downarrow}$	.307$^{\downarrow}$	.802$^{\downarrow}$	.455$^{\downarrow}$	.836$^{\downarrow}$	.620$^{\downarrow}$
Conc-SR	.752$^{\downarrow}$	.438$^{\downarrow}$	.679$^{\downarrow}$	.333$^{\downarrow}$	.672$^{\downarrow}$	**.311**	.671$^{\downarrow}$	.308	.839$^{\downarrow}$	.501$^{\downarrow}$	.875$^{\downarrow}$	.702$^{\downarrow}$
SmVR-SVM	**.805**	**.453**	**.727**	**.353**	**.681**	**.311**	**.694**	**.316**	**.866**	**.532**	**.901**	**.727**

SmVC-SVM**:** semi-supervised multi-view classification. Classification counterpart to our ranking approach, iteratively labeling examples based on the consensus of classifiers[3] trained on each view [3].

SmVR-SVM**:** semi-supervised multi-view ranking (this paper). Combines the multiple views available in the training data, using both the labeled and the unlabeled examples as described in Algorithm 1.

All experiments use linear kernels, SVM regularization parameters are set to the default values,[4] and S (inner loop of Algorithm 1) is set to 15000.[5] Performance is reported in terms of average AUC and AvP of the view-specific scoring functions over the five languages.

5.2 Results

We first compare the performance of all the models and investigate the effect of the various aspects of our model. We compute the Average Precision and AUC of the models obtained using 10 labeled training examples, and the unlabeled training examples (for the models that use them). We chose 10 labeled examples in order to study the role of unlabeled data in the regime where very little annotation is available. Table 2 summarizes the results obtained on our six topics by the approaches described above, averaged over five languages and repeated over 10 random training/test splits. Bold face indicates the highest performance, and ↓ indicates that the performance is significantly worse than the best result, according to a Wilcoxon rank sum test used at $p < 0.01$ [19]. These results show that our approach, SmVR-SVM, consistently and significantly outperforms the four competing models. Unsurprisingly, the simple baseline sVR-SVM yields the lowest performance. The results clearly show that all semi-supervised strategies outperform that baseline, suggesting that the unlabeled data already significantly improves the performance in both AUC and AvP. However, comparing the semi-supervised rankers shows that the multiview (SmVR-SVM) learning

[3] linear SVMs minimizing misclassification error using SVM-Perf [17].

[4] With little annotation, cross-validation proved unreliable to tune hyperparameters.

[5] Increasing this value has almost no influence on the results.

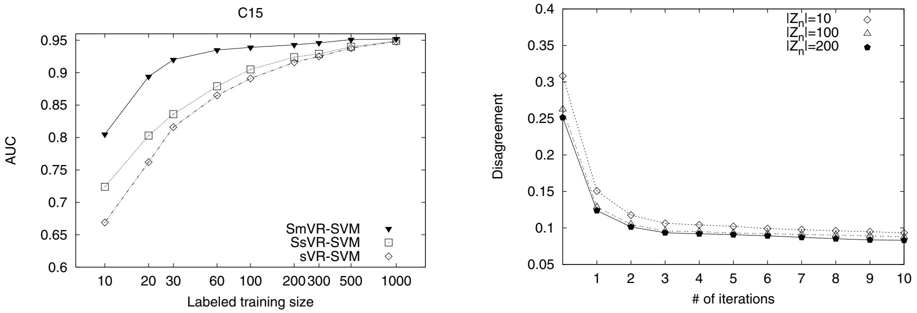


Fig. 1. Left: AUC learning curves for topic C15, averaged over 10 randomly chosen training/test splits. **Right:** Disagreement, averaged over topics, as training progresses, for 10, 100 and 200 labeled examples.

framework brings more performance increase (up to 16 points in AvP for M11) than the single view (SsVR-SVM) algorithm.

Table 2 also shows that our ranking approach clearly and consistently outperforms semi-supervised multiview classification (SmVC-SVM). This suggests that the classification approach is not well suited to solving a bipartite ranking problem even with the help of the richer, multiview information. In fact, this is reinforced by the fact that, in our experiments, even single-view ranking (SsVR-SVM) outperforms multiview classification (SmVC-SVM). Note also that our approach also outperforms Conc-SR, showing that using all views through a simple concatenation is not as efficient as a proper multiview framework.

We now investigate several issues. First, we study the evolution of the performances of our approach depending on the training set size. Then, we show how our approach effectively minimizes the disagreement of the view-specific rankers on the unlabeled data. We also study the influence on the performance of the imbalance of the initial labeled training set, and finally, we investigate the difference between our multiview algorithm and the approach consisting of simply learning on the concatenated views, for increasing numbers of views.

Effect of the Labeled Training Set Size: One of the motivations for using semi-supervised learning is that labeled data is usually costly to acquire. It is therefore of great interest to investigate how the performance of the various algorithms evolve as the number of available labeled examples changes. Figure 1 illustrates this on class C15 (other classes are qualitatively similar). It shows how the AUC evolves with the number of labeled documents in the initial training set. Our experiments correspond to $|Z_n| = 10$, the leftmost points on the curves in the figure. As expected, performance increases monotonically with additional labeled data. The relative ordering observed in Table 2 is maintained throughout the entire range of labeled data, with SmVR-SVM performing consistently (but diminishingly) better than SsVR-SVM, which in turn does better than sVR-SVM.

When there are enough labeled examples, all algorithms actually converge to the same AUC value, suggesting that the labeled data carries sufficient information and that no *additional* information could be extracted from the multiview

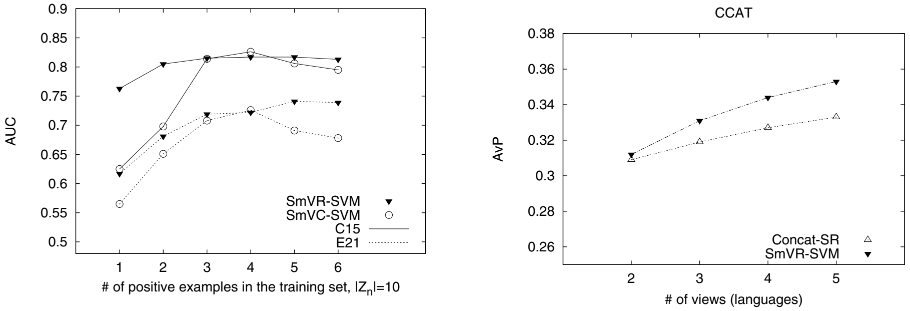


Fig. 2. Left: AUC vs. number of positive examples in the 10 labeled training documents, on two topics, for ranking vs. classification. **Right:** AvP vs. # of views/languages on topic CCAT for multiview learning (SmVR-SVM) vs. concatenation (Conc-SR).

unlabeled examples. For low amounts of labeled training data, however, the contribution of the unlabeled data used in semi-supervised learning is clear.

Evolution of the Disagreement: The motivation of our algorithm is that we improve the generalization performance by minimizing the disagreement between the rankers trained on the different views. Figure 1 shows how the disagreement on unlabeled examples (Eq. 4), averaged over all topics, evolves during training. At iteration 0, the disagreement corresponds to that of sVR-SVM (the disagreement of models trained independently on each view without using any unlabeled data). The figure shows that for all three training set sizes pictured, as learning progresses, the disagreement decreases towards a small asymptotic limit. Having more labeled examples helps start with a lower disagreement. However, even for 10 labeled instances, multiview learning brings the disagreement well below that observed for the single-view approach with 20 times more labeled data.

Effect of the Number of Positive Examples: The results in Table 2 use labeled training sets respecting the real class proportions. The learning tasks were thus extremely difficult, since very few positive examples were available. As ranking costs are supposed to be more immune to class imbalance than the misclassification error, we investigate how the performance of the classification versus ranking approaches evolve when more positive examples are available as initial labeled training data. Figure 2 compares the performance of the multiview ranking (SmVC-SVM) and classification (SmVR-SVM) algorithms for increasing numbers of positive examples. We picked two classes, C15 and E21, which yield differing patterns of results in Table 2: the impact of our method is much larger for C15 than it is for E21. In both cases, Figure 2 shows that as the proportion of positive examples nears 50% (5 positive examples), the classification approach becomes more competitive, while the multiview ranking algorithm appears a lot more robust to class imbalance. In fact, when the number of positive examples grows past 50% (rightmost edge of the graph, 6 positive and 4 negatives), the performance of the classification approach starts decreasing again.

Comparison to Concatenated Views: The weighted pseudo-labeling step of our algorithm uses a unanimous decision over the views to select examples that should be added to the training set. The unanimous vote is used as a confidence measure, in order to avoid introducing too much noise at each iteration. One may then ask how the performance of the multiview approach evolves depending on the number of available languages? Figure 2 plots the `AvP` observed for topic `CCAT`, as a function of the number of available languages, for our algorithm `SmVR-SVM` and for the semi-supervised single view model which uses the concatenation of the views, `Conc-SR`. Results for less than 5 languages are averaged over all possible subsets of languages. The results show that the performance of `SmVR-SVM` and `Conc-SR` increase as more views are available, with a growing advantage for the multiview approach as the number of languages increases. This confirms that the multiview paradigm offers a better semi-supervised learning principle than the single view learning, and is better able to leverage the additional information available in the different view than simple concatenation of the inputs.

6 Conclusion

We presented an algorithm for bipartite ranking with unlabeled data and multiple views, and showed its empirical performance on a multilingual data collection. The algorithm exhibits better ranking performance than both single-view semi-supervised ranking and multiview classification, in particular when the initial labeled training set is highly unbalanced. Our analysis and algorithms are tailored to bipartite ranking. This allowed us to give experimental comparisons with semi-supervised classification algorithms and existing semi-supervised single view ranking algorithms for bipartite ranking. The results show the importance of optimizing a ranking criterion, as well as the relative performances of single view and multiview approaches.

A direct extension of our work is to examine the possibility of multiview, semi-supervised ranking when the reference ranking information is not bipartite, but take the form of either scores on an ordinal scale, or more generally preference relations. Indeed, even though the weighted pseudo-labeling step is specific to bipartite ranking, the learning principle of Section 3, as well as the method for selecting pairs in the algorithm, extend to more general cases in a straightforward manner. Another direction is to extend our method to search problems, where the goal is to infer rankings on a fixed collection depending on a user query.

References

1. Agarwal, S., Graepel, T., Herbrich, R., Har-Peled, S., Roth, D.: Generalization bounds for the area under the roc curve. JMLR 6, 393–425 (2005)
2. Amini, M.R., Truong, T.V., Goutte, C.: A boosting algorithm for learning bipartite ranking functions with partially labeled data. In: SIGIR 2008 (2008)
3. Amini, M.R., Usunier, N., Goutte, C.: Learning from multiple partially observed views – an application to multilingual text categorization. In: NIPS-22 (2009)

4. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: ICML 2004 (2004)
5. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 15, 1373–1396 (2003)
6. Belkin, M., Niyogi, P.: Semi-supervised learning on riemannian manifolds. Machine Learning 56, 209–239 (2004)
7. Blum, A., Mitchell, T.M.: Combining labeled and unlabeled data with co-training. In: COLT, pp. 92–100 (1998)
8. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press, Cambridge (2006)
9. Clémençon, S., Lugosi, G., Vayatis, N.: Ranking and scoring using empirical risk minimization. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 1–15. Springer, Heidelberg (2005)
10. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research 10, 243–270 (1999)
11. Cortes, C., Mohri, M.: AUC optimization vs. error rate minimization. In: NIPS-16 (2003)
12. Diethe, T., Hardoon, D.R., Shawe-Taylor, J.: Multiview fisher discriminant analysis. In: NIPS Workshop on Learning from Multiple Sources (2008)
13. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. JMLR 4, 933–969 (2003)
14. Hoi, S., Jin, R.: Semi-supervised ensemble ranking. In: Proceedings of the 23rd National Conference on Artificial intelligence, pp. 634–639 (2008)
15. Joachims, T.: Optimizing search engines using clickthrough data. In: KDD (2002)
16. Joachims, T.: A support vector method for multivariate performance measures. In: ICML 2005, pp. 377–384 (2005)
17. Joachims, T.: Training linear svms in linear time. In: KDD 2006, pp. 217–226 (2006)
18. Kakade, S.M., Foster, D.P.: Multi-view regression via canonical correlation analysis. In: Bshouty, N.H., Gentile, C. (eds.) COLT. LNCS (LNAI), vol. 4539, pp. 82–96. Springer, Heidelberg (2007)
19. Lehmann, E.: Nonparametric Statistical Methods Based on Ranks. McGraw-Hill, New York (1975)
20. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
21. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using em. Machine Learning 39, 103–134 (2000)
22. Ralaivola, L.: Semi-supervised bipartite ranking with the normalized Rayleigh coefficient. In: ESANN 2009 (2009)
23. Robertson, S., Callan, J.: Routing and filtering. In: TREC: Experiment and Evaluation in Information Retrieval, ch. 5, pp. 99–121. MIT Press, Cambridge (2005)
24. Rosenberg, D.S., Bartlett, P.L.: The Rademacher complexity of co-regularized kernel classes. JMLR, 396–403 (2007)
25. Sridharan, K., Kakade, S.M.: An information theoretic framework for multi-view learning. In: COLT 2008, pp. 403–414 (2008)
26. Usunier, N., Amini, M.-R., Gallinari, P.: A data-dependent generalisation error bound for the auc. In: Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning (2005)
27. Zhu, X.: Semi-supervised learning literature survey. Technical report, Carnegie Mellon University Department of Computer Sciences (2006)

Non-redundant Subgroup Discovery in Large and Complex Data

Matthijs van Leeuwen[1] and Arno Knobbe[2]

[1] Dept. of Information & Computing Sciences, Universiteit Utrecht, The Netherlands
[2] LIACS, Universiteit Leiden, The Netherlands
mleeuwen@cs.uu.nl, knobbe@liacs.nl

Abstract. Large and complex data is challenging for most existing discovery algorithms, for several reasons. First of all, such data leads to enormous hypothesis spaces, making exhaustive search infeasible. Second, many variants of essentially the same pattern exist, due to (numeric) attributes of high cardinality, correlated attributes, and so on. This causes top-k mining algorithms to return highly redundant result sets, while ignoring many potentially interesting results.

These problems are particularly apparent with Subgroup Discovery and its generalisation, Exceptional Model Mining. To address this, we introduce *subgroup set mining*: one should not consider individual subgroups, but sets of subgroups. We consider three degrees of redundancy, and propose corresponding heuristic selection strategies in order to eliminate redundancy. By incorporating these strategies in a beam search, the balance between exploration and exploitation is improved.

Experiments clearly show that the proposed methods result in much more diverse subgroup sets than traditional Subgroup Discovery methods.

1 Introduction

In this paper, we assume that we are dealing with complex data. This complexity can be due to several aspects of the data, e.g. datasets may contain many rows as well as many attributes, and these attributes may be of high cardinality. Such complex data is challenging for existing discovery algorithms, primarily for reasons of computation time: all these aspects will have an impact on the time required for mining the data. Especially where numeric data is concerned, detailed analysis of the data will imply high cardinalities on such attributes, and many candidate hypotheses will need to be tested. Also, complexity may reside in the discovery task, for example when modelling non-trivial interactions between attributes. The result of these challenges is that individual candidate testing becomes very time-consuming, and hypothesis spaces become prohibitively large.

In the majority of discovery algorithms, including those for Subgroup Discovery (SD) [6,17], it is assumed that complete solutions to a particular discovery task are required, and thus some form of exhaustive search is employed. In order to obtain efficiency, these algorithms typically rely on top-down search combined

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 459–474, 2011.

with considerable pruning, exploiting either anti-monotonicity of the quality measure (e.g. frequency), or so-called optimistic estimates of the maximally attainable quality at every point in the search space [3]. With small datasets and simple tasks, these tricks work well and give complete solutions in reasonable time. However, on the complex datasets that we assume, exhaustive approaches simply become infeasible, even when considerable pruning can be achieved. Additionally, we consider Exceptional Model Mining (EMM) [11,10], which allows multiple target attributes and complex models to be used for measuring quality. With EMM in particular, we are often dealing with quality measures that are not monotonic, and for which no optimistic estimates are available.

Apart from the computational concerns with discovery in large datasets, one also needs to consider the practicality of complete solutions in terms of the size of the output. Even when using condensed representations [13,14] or some form of pattern set selection [1,7,15] as a post-processing step, the end result may still be unrealistically large, and represent tiny details of the data overly specifically. The experienced user of discovery algorithms will recognise the large level of redundancy that is common in the final pattern set. This redundancy is often the result of dependencies between the (non-target) attributes, which lead to large numbers of variations of a particular finding. Note that large result sets are problematic even in top-k approaches. Large result sets are obviously not a problem in top-1 approaches, but they are when $k \geq 2$, as the mentioned dependencies will lead to the top of the pattern list being populated with different variations on the same theme, and alternative patterns dropping out of the top-k. This problem is aptly illustrated by Figure 1, which shows that the top-100 subgroups obtained on *Credit-G* cover almost exactly the same tuples.

Approach and contributions. The obvious alternative to exhaustive search, and the one we consider in this paper, is of course *heuristic search*: employ educated guesses to consider only that fraction of the search space that is likely to contain the patterns of interest. When performing heuristic search, it is essential to achieve a good balance between *exploitation* and *exploration*. In other words, to focus and extend on promising areas in the search space, while leaving room for several alternative lines of search. In this work, we will implement this balance by means of *beam search*, which provides a good mixture between parallel search (exploration) and hill-climbing (exploitation). Within the beam search framework, we will experiment with different variations of achieving diversity in the *beam*, that is, the current list of candidates to be extended. Due to the above-mentioned risk of redundancy with top-k selection, the level of exploration

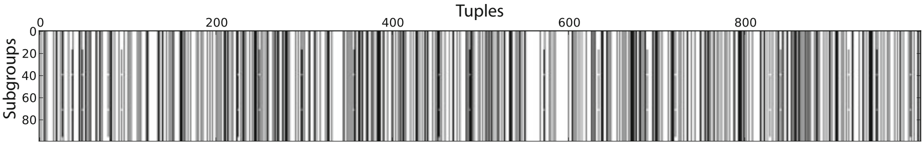


Fig. 1. Redundancy in top-k Subgroup Discovery. Shown are the covers (in *black*) of the top-100 subgroups obtained on *Credit-G* with weighted relative accuracy.

within a beam can become limited, which will adversely affect the quality of the end result. Inspiration for selecting a diverse collection of patterns for the beam at each search level will come from pattern set selection techniques, which were originally designed for post-processing the end-result of discovery algorithms.

In Section 2, we will first formalise both Subgroup Discovery and Exceptional Model Mining, after which we will recap the commonly used search techniques, including the standard beam search algorithm. We will then introduce the notion of *subgroup set mining* in Section 3, and argue that it is better to mine subgroup sets rather than individual subgroups, to ensure diversity. This leads to the *Non-Redundant Generalised Subgroup Discovery* problem statement. We will show that redundancy in subgroup sets can be formalised in (at least) three different ways, each subsequent definition being more strict than its predecessor. Each of these three degrees of redundancy is used as basic principle for a beam selection strategy in Section 5. Section 4 presents the quality measures that will be used in the experiments, which are presented in Section 6. We round up with related work and conclusions in Sections 7 and 8.

2 Preliminaries

2.1 Subgroup Discovery and Exceptional Model Mining

We assume that the tuples to be analysed are described by a set of attributes A, which consists of k *description attributes* D and l *model (or target) attributes* M ($k \geq 1$ and $l \geq 1$). In other words, we assume a supervised setting, with at least a single target attribute M_1 (in the case of classical SD), but possibly multiple attributes $M_1, \ldots, M_l$ (in the case of EMM). Each attribute D_i (resp. M_i) has a domain of possible values $\mathrm{Dom}(D_i)$ (resp. $\mathrm{Dom}(M_i)$). Our dataset $\mathcal{S}$ is now a bag of tuples t over the set of attributes $A = \{D_1, \ldots, D_k, M_1, \ldots, M_l\}$. We use x^D resp. x^M to denote the projection of x onto its description resp. model attributes, e.g. $t^D = \pi_D(t)$ in case of a tuple, or $\mathcal{S}^M = \pi_M(\mathcal{S})$ in case of a bag of tuples. Equivalently for individual attributes, e.g. $\mathcal{S}^{M_i} = \pi_{M_i}(\mathcal{S})$.

Arguably the most important concept in this paper is the *subgroup*, which consists of a *description* and corresponding *cover*. A *subgroup (cover)* is a bag of tuples $G \subseteq \mathcal{S}$ and $|G|$ denotes its size, also called *subgroup size* or *coverage*.

A *subgroup description* is an indicator function s, as a function of description attributes D. That is, it is a function $s : (\mathrm{Dom}(D_1) \times \ldots \times \mathrm{Dom}(D_k)) \mapsto \{0, 1\}$, and its corresponding subgroup cover is $G_s = \{t \in \mathcal{S} \mid s(t^D) = 1\}$. As is usual, in this paper a subgroup description is a *pattern*, consisting of a conjunction of conditions on the description attributes, e.g. $D_x = \mathit{true} \wedge D_y \leq 3.14$. Such a pattern implies an indicator function as just defined.

Given a subgroup G, we would like to know how interesting it is, looking only at its model (or target) data G^M. We quantify this with a quality measure. A *quality measure* is a function $\varphi : \mathcal{G}^M \mapsto \mathbb{R}$ that assigns a numeric value to a subgroup $G^M \subseteq \mathcal{S}^M$, with $\mathcal{G}^M$ the set of all possible subsets of $\mathcal{S}^M$.

Subgroup Discovery and Exceptional Model Mining. The above definitions allow us to define the two main variations of data mining tasks that feature in this paper: Subgroup Discovery (SD) and Exceptional Model Mining (EMM). As mentioned, in SD we consider datasets where only a single model attribute M_1 (the target) exists. We are interested in finding the top-ranking subgroups according to a quality measure φ that determines the level of interestingness in terms of unusual distribution of the target attribute M_1:

Problem 1 (Top-k Subgroup Discovery). Suppose we are given a dataset $\mathcal{S}$ with $l = 1$, a quality measure φ and a number k. The task is to find the k top-ranking subgroups $\mathcal{G}_k$ with respect to φ.

EMM is a generalisation of the well-known SD paradigm, where the single target attribute is replaced by a collection of model attributes [11]. Just like in SD, EMM is concerned with finding subgroups that show an unusual distribution of the model attributes. However, dependencies between these attributes may occur, and it is therefore desirable to consider the joint distribution over $M_1, \ldots, M_l$. For this reason, modelling over G^M is employed to compute a value for φ. If the model induced on G^M is substantially different from the model induced on $\mathcal{S}^M$, quality is high and we call this an *exceptional model*. We can now formally state the EMM problem.

Problem 2 (Top-k Exceptional Model Mining). Suppose we are given a dataset $\mathcal{S}$, a quality measure φ and a number k. The task is to find the k top-ranking subgroups $\mathcal{G}_k$ with respect to φ.

2.2 Subgroup Search

To find high-quality subgroups, the usual choice is a top-down search strategy. The search space is traversed by starting with simple descriptions and refining these along the way, from general to specific. For this a *refinement operator* that specialises subgroup descriptions is needed. A *minimum coverage* threshold (*mincov*) is used to ensure that a subgroup covers at least a certain number of tuples. A *maximum depth* (*maxdepth*) parameter imposes a maximum on the number of conditions a description may contain.

Exhaustive search. When exhaustive search is possible, depth-first search is commonly used. This is often the case with moderately sized nominal datasets with a single target. Whenever possible, (anti-)monotone properties of the quality measure are used to prune parts of the search space. When this is not possible, so-called *optimistic estimates* can be used to restrict the search space. An optimistic estimate function computes the highest possible quality that any refinement of a subgroup could give. If this upper bound is lower than the quality of the current kth subgroup, this branch of the search space can be safely ignored.

Beam search. When exhaustive search is not feasible, beam search is the widely accepted heuristic alternative. It also uses a levelwise top-down strategy and the same refinement operator, but it explores only part of the search space. The basic algorithm is shown as Algorithm 1. On each level, the w highest ranking subgroups with respect to quality are selected for the *beam*. Candidate subgroups for the next level are generated from individual subgroups b using the refinement

Algorithm 1. Beam Search

```
Input: A dataset $\mathcal{S}$, a quality measure $\varphi$ and parameters $k$, $w$, $mincov$ and $maxdepth$.
Output: $\mathcal{R}$, an approximation of the top-$k$ subgroups $\mathcal{G}_k$.
$\mathcal{R} \leftarrow \emptyset$, $Beam \leftarrow \{\emptyset\}$, $depth = 1$
while $depth \leq maxdepth$ do
  $Cands \leftarrow \emptyset$
  for all $b \in Beam$ do
    $Cands \leftarrow Cands \cup \text{GenerateRefinements}(b, mincov)$
  for all $c \in Cands$ do
    $\text{UpdateTopK}(\mathcal{R}, k, c, \varphi(c))$
  $Beam \leftarrow \text{SelectBeam}(Cands, w, \varphi)$
  $depth \leftarrow depth + 1$
return $\mathcal{R}$
```

operator (*GenerateRefinements*), while respecting the *mincov* parameter. The initial candidate set is generated from the empty subgroup description. *SelectBeam* selects the w highest ranking $c \in Cands$ (with respect to φ) to form the beam for the next level.

3 Non-Redundant Generalised Subgroup Discovery

The redundancy issues experienced with SD/EMM algorithms suggest that we should not only look at each individual subgroup locally, but also take the other subgroups into account. That is, we should consider *subgroup set mining*, similar to recent pattern set selection approaches [1,7,15].

Problem 3 (Non-Redundant Generalised Subgroup Discovery). Suppose we are given a dataset $\mathcal{S}$, a quality measure φ and a number k. The task is to find a non-redundant set $\mathcal{G}$ of k high-quality subgroups.

The term *Generalised Subgroup Discovery* is used to emphasise that it encompasses both SD and EMM.

Although it may be clear to the data miner whether a (small) set of patterns contains redundancy or not, formalising redundancy is no trivial task. We can consider three degrees of redundancy removal.

In a *non-redundant subgroup set* $\mathcal{G}$, all pairs $G_i, G_j \in \mathcal{G}$ (with $i \neq j$) should have substantially different:

1. *subgroup descriptions*, or
2. *subgroup covers*, or
3. *exceptional models.* (Only in the case of EMM.)

Note that each subsequent degree is more strict than its predecessor. On the first, least restrictive degree, substantially different descriptions are allowed, ignoring any potential similarity in the cover. The second degree of redundancy would also address this kind of similarity in the subgroup covers. The third degree of redundancy will consider subgroups that are different in both description and cover, and will address their difference in terms of the associated models built on the model attributes M.

In Section 5, each of the three degrees of redundancy will be used as basic principle for a subgroup set selection method and be incorporated in the beam search. The resulting search strategies eliminate redundancy in subgroup sets.

To quantify redundancy in subgroup sets, we consider the subgroup covers because it is independent of any other choices and can be easily interpreted. By assuming a uniform distribution of all subgroup covers over all tuples in the dataset, we can compute an *expected cover count* and measure how far each individual tuple's cover count deviates from this. This results in the following.

Definition 1 (Cover Redundancy). *Suppose we are given a dataset $\mathcal{S}$ and a set of subgroups $\mathcal{G}$. Define the* cover count *of a tuple $t \in \mathcal{S}$ as $c(t, \mathcal{G}) = \sum_{G \in \mathcal{G}} s_G(t)$. The* expected cover count *$\hat{c}$ of a random tuple $t \in \mathcal{S}$ is defined as $\hat{c} = \frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} c(t, \mathcal{G})$. The* Cover Redundancy *CR is now computed as:*

$$CR^{\mathcal{S}}(\mathcal{G}) = \frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} \frac{|c(t, \mathcal{G}) - \hat{c}|}{\hat{c}}$$

The larger the CR, the larger the deviation from the uniform distribution. Because Generalised Subgroup Discovery aims to find only those parts of the data that stand out, this measure on itself does not tell us much. However, if we have several subgroup sets of (roughly) the same size and for the same dataset, a lower CR indicates that less tuples are covered by more subgroups than expected, and thus the subgroup set is more diverse/less redundant.

As an example, the subgroup set in Figure 1 has a CR of 1.19. Clearly, this cover distribution is highly undesirable and (much) lower values are preferred.

4 Quality Measures

Weighted Relative Accuracy. Weighted Relative Accuracy (*WRAcc*) [9] is a well-known SD quality measure for datasets with one binary target attribute. Let 1^G (resp. $1^{\mathcal{S}}$) denote the fraction of ones in the target attribute, within the subgroup (resp. entire dataset). Weighted Relative Accuracy is then defined as $\varphi_{WRAcc}(G) = \frac{|G|}{|\mathcal{S}|}(1^G - 1^{\mathcal{S}})$.

Weighted Kullback-Leibler divergence. We previously [10] introduced a measure based on the Kullback-Leibler (KL) divergence. Each attribute-value is assumed to be an independently drawn sample from an underlying random variable. The empirical probability distribution for attribute M_i is estimated by $\hat{P}$. We here present an alternative that weighs quality by subgroup size, because this works better in combination with a levelwise search (without this weight, smaller subgroups always tend to have larger qualities). This measure can be used with either a single or multiple binary model attributes, and even with nominal attributes.

Definition 2 (WKL quality). *Given a database $\mathcal{S}$ and subgroup G, define (independent)* Weighted KL quality *as*

$$\varphi_{\mathrm{WKL}}(G^M) = \frac{|G|}{|\mathcal{S}|} \sum_{i=1}^{l} \mathrm{KL}(\hat{P}(G^{M_i}) \parallel \hat{P}(\mathcal{S}^{M_i}))$$

Weighted Krimp Gain. In [10] we introduced a second measure that, contrary to (Weighted) KL quality, does take associations between (binary) attributes into account. It uses Krimp code tables [16] as models, but the principle is equivalent to that of *WKL*: a subgroup is interesting if it can be compressed much better by its own compressor, than by the compressor induced on the overall database. Similar to *WKL* quality, we here introduce a weighted alternative.

Definition 3 (Weighted Krimp Gain). *Let $\mathcal{D}$ be a binary database, $G \subseteq \mathcal{D}$ a subgroup, and $CT_{\mathcal{D}}$ and CT_G their respective optimal code tables. We define the* Weighted Krimp Gain *of group G from $\mathcal{D}$, denoted by* $\mathrm{WKG}(G \parallel \mathcal{D})$*, as*

$$\mathrm{WKG}(G \parallel \mathcal{D}) = L(G \mid CT_{\mathcal{D}}) - L(G \mid CT_G),$$

with $L(G \mid CT)$ the size of G, in bits, encoded with code table CT.

Given this, defining the quality measure is straightforward.

Definition 4 (WKG quality). *Let $\mathcal{S}$ be a database and $G \subseteq \mathcal{S}$ a subgroup. Define* Weighted KG quality *as* $\varphi_{\mathrm{WKG}}(G^M) = \mathrm{WKG}(G^M \parallel \mathcal{S}^M)$.

5 Non-Redundant Beam Selection

In this section we show how selection strategies based on the three degrees of redundancy from Section 3 can be incorporated in the basic beam search algorithm (see Algorithm 1). Instead of simply choosing the –potentially highly redundant– top-k subgroups for the beam, we will modify the algorithm to select diverse *subgroup sets* at each level. In other words, we strive to achieve high-quality yet non-redundant beam selection.

A *beam selection strategy* is a selection scheme that decides which candidates are included in the beam, and is invoked by *SelectBeam* in Algorithm 1. We will refer to regular top-k beam selection as the *Standard* strategy.

Most pattern set selection criteria require all possible pattern sets to be taken into consideration to ensure that the global optimum is found. However, large numbers of subgroups may be evaluated at each search level and such exhaustive strategies are therefore infeasible. Hence, we have to resort to greedy and heuristic methods, as is usual in pattern set selection [1,7,15]. The following three selection strategies correspond to the three degrees of redundancy.

Description-based beam selection. Order all candidates descending by quality and consider them one by one until beam width w is reached. For each considered subgroup $G \in Cands$, discard it if its quality and all but 1 conditions are equal to that of any $b \in Beam$, otherwise include it in the beam. Time complexity for selecting the beam of a single level is $\mathcal{O}(|Cands| \cdot log(|Cands|) + |Cands| \cdot depth)$ (the current search *depth* influences how long a comparison of descriptions takes).

Cover-based beam selection, This strategy focuses on the subgroup covers and how they overlap. A score based on multiplicative weighted sequential covering [9] is used to weigh the quality of each subgroup, aiming to minimise the overlap between the selected subgroups. This score is defined as

$$\Omega(G, Beam) = \frac{1}{|G|} \sum_{t \in G} \alpha^{c(t,Beam)},$$

where $\alpha \in \langle 0, 1]$ is the weight parameter. The less often tuples in subgroup G are already covered by subgroups in the beam, the larger the score. If the cover contains only previously uncovered tuples, $\Omega(G, Beam) = 1$.

In w iterations, w subgroups are selected for inclusion in the beam. In each iteration, the subgroup that maximises $\Omega(G, Beam) \cdot \varphi(G)$ is selected. The first selected subgroup is always the one with the highest quality, since the beam is empty and $\Omega(G, Beam) = 1$ for all G. After that, the Ω-scores for the remaining *Cands* are updated each iteration. Complexity per level is $\mathcal{O}(w \cdot |Cands| \cdot |\mathcal{S}|)$.

Compression-based beam selection. To be able to do model-based beam selection, a (dis)similarity measure on models is required. For this purpose, we focus on the models used by the *WKL* and *WKG* quality measures. These measures have in common that they rely on *compression*; they assume a coding scheme and the induced models can therefore be regarded as *compressors*.

In case of *WKG*, the compressor is the code table induced by Krimp. In case of *WKL*, the compressor replaces each attribute-value x by a code of optimal length $L(x)$ based on its marginal probability, i.e. $L(x) = -\log_2(\hat{P}(M_i = x))$.

Adopting the MDL philosophy [4], we say that the best set of compressors is that set that together compresses the dataset best. Selecting a set of compressors is equivalent to selecting a set of subgroups, since each subgroup has exactly one corresponding compressor. Since exhaustive search is infeasible, we propose the following heuristic.

1. We start with the 'base' compressor that is induced on the entire dataset, denoted C^S. Each $t \in S$ is compressed with this compressor, resulting in encoded size $L(S \mid C^S)$.
2. Next, we iteratively search for the subgroup that improves overall compression most, relative to the compression provided by the subgroups already selected. That is, the first selected subgroup is always the top-ranked one, since its compressor C^1 gives the largest gain with respect to $L(S \mid C^S)$.
3. Each transaction is compressed by the last picked subgroup that covers it, and by C^S if it is not yet covered by any. So, after the first round, part of the transactions are encoded by C^S, others by C^1.

4. Assuming this encoding scheme, select that subgroup $G \in Cands \setminus \{C^1, \dots\}$ that maximises $L(S \mid C^S, C^1, \dots) - L(S \mid C^S, C^1, \dots, G)$ in each subsequent step. Stop when the beam has attained its desired width w.

To perform this selection strategy, all compressors belonging to the subgroups of a certain level are required. If these can be kept in memory, the complexity of the selection scheme is $\mathcal{O}(w \cdot |Cands| \cdot |\mathcal{S}| \cdot |M|)$, where $|M|$ is the number of model attributes. However, keeping all compressors in memory may not be possible. They could then be either cached on disk or reconstructed on demand, but both approaches would severely impact runtimes.

Each subsequent beam selection strategy is more strict than its predecessor, but also computationally more demanding. This offers the data miner the opportunity to trade-off diversity with computation time.

5.1 Improving Individual Subgroups

Despite all efforts to prevent and eliminate redundancy in the result set, some of the found subgroups may be overly specific. This may be caused by a large search depth, but also by heuristic choices in e.g. the refinement operator. For example, the subgroup corresponding to $A = true \wedge B = true$ might have the highest possible quality, but never be found since neither $A = true$ nor $B = true$ has high quality. However, $C = false \wedge A = true \wedge B = true$ could be found. Now, *pruning* the first condition would give the best possible subgroup.

We propose to improve individual subgroups by pruning the subgroup descriptions as a post-processing step, based on the concept of *dominance*. A subgroup G_i *dominates* a subgroup G_j iff

1. the conditions of the description of G_i are a strict subset of those of G_j, and
2. the quality of G_i is higher than or equal to that of G_j, i.e. $\varphi(G_i) \geq \varphi(G_j)$.

Observe that although dominance is clearly inspired by relevancy [2], it is not the same. The former is more generic, making it also suitable for e.g. EMM.

The heuristic method we propose for *dominance-based pruning* is to consider each of the conditions in a subgroup description one by one, in the order in which they were added. If removing a condition does not decrease the subgroup's quality, then permanently remove it, otherwise keep it.

5.2 Non-Redundant Beam Search

The overall subgroup set mining process we propose consists of three steps. First, a beam search (Algorithm 1) is performed to mine N subgroups, with any of the proposed beam selection strategies (plugged in as *SelectBeam*). Next, each of the N resulting subgroups is individually pruned based on dominance, and syntactically equivalent subgroups are removed. As the final result set potentially also suffers from the redundancy problems of top-k-selection, a selection strategy is used to select S subgroups ($S \ll N$) from the remaining subgroups ('post-selection'). For this, the same strategy as during the beam search is used.

Refinements. We distinguish three types of description attributes, each with its own associated condition types: $\{=\}$ for binary, $\{=, \neq\}$ for nominal and $\{<, >\}$ for numeric attributes. For binary and nominal attributes, the refinement operator always generates all possible refinements, i.e. each combination of condition type and attribute-value. To prevent the search space from exploding, the values of a numeric attribute are locally binned into 6 equal-sized bins and $\{<, >\}$-conditions are generated for the 5 split points obtained this way. This 'on-the-fly' discretisation, performed upon subgroup refinement, results in a much more fine-grained binning than 'a priori' discretisation of numeric attributes.

Except for refinements that lead to a contradiction, all refinements for all description attributes are always considered. (Adding $D_x = true$ to a description that already contains $D_x = false$ would be meaningless, for example.) Consequently, multiple conditions on the same attribute can be imposed; especially with nominal attributes, slowly peeling off tuples with $\neq$ can be helpful.

6 Experiments

Datasets. To evaluate the proposed methods, we perform experiments on the datasets listed in Table 1. The upper three datasets, taken from the UCI repository[1], contain a single target (SD), the lower four datasets have multiple model attributes (EMM). Two variants of the UCI *Adult* dataset are used: *Adult-SD* is the commonly used variant, with the binary class label as single target, in *Adult-EMM* all numeric attributes are considered as description attributes, and all binary attributes as model attributes (except for class, which is not used). Furthermore, we take the *Emotions* and *Yeast* datasets from the 'Mulan' repository[2], and we use the *Mammals* dataset [5] (each of these has numeric description attributes and binary model attributes).

Table 1. Datasets. For each dataset the number of tuples, the number of description and model attributes, and the *minsup* used for *WKG* are given.

Dataset	*Properties*			*WKG*
	$\mid S \mid$	$\mid D \mid$	$\mid M \mid$	minsup
Adult-SD	48842	105	1	-
Credit-G	1000	20	1	-
Mushroom	8124	22	1	-
Adult-EMM	48842	6	99	10%
Emotions	593	72	6	1%
Mammals	2221	67	124	-
Yeast	2417	103	14	1%

Methods for comparison. Depth-first search (DFS) is used, with *WRAcc* in combination with tight optimistic estimate [3]. With DFS, only a single condition per attribute is allowed and all attributes are considered in a fixed order. This is necessary to limit the size of the search space and thus computation time, but also means that beam search can potentially reach better solutions.

[1] `http://archive.ics.uci.edu/ml/`

[2] `http://mulan.sourceforge.net/datasets.html`

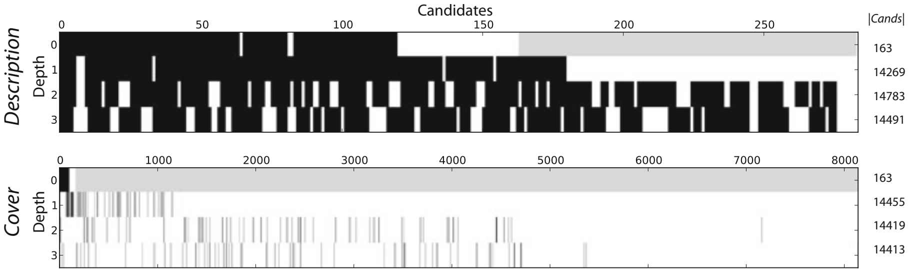


Fig. 2. Two beam selection strategies in action: description-based and cover-based. For each level in the beam search, it is shown which candidate subgroups are selected for inclusion in the beam (*black*) and which are ignored (*white*). Candidates are ordered descending on quality. On the right, the total number of candidate subgroups for each level is shown (candidates not shown are not selected). *Credit-G* with *WRAcc*.

An often adopted approach to mining pattern sets is the 2-step approach, where 1) all patterns are mined and 2) a subset of these patterns is selected as post-processing step. We test this approach by first using DFS or standard beam search to mine the top-N subgroups, and then use cover-based selection to select S subgroups from this (denoted '+PS', for post-selection).

Search parameters. In all experiments, $N = 10,000$ subgroups are mined, from which $S = 100$ are selected for the final subgroup set. A maximum depth $maxdepth = 5$, minimum coverage $mincov = 10$, and beam width $w = 100$ are used. Preliminary experiments showed that changing these parameters has the same effect on all search strategies, keeping their differences intact. Since our aim is to compare the different strategies, we keep these fixed. Weight parameter α for cover-based beam selection is set to 0.9, since preliminary experiments indicated that this gives a good balance between quality and cover diversity.

6.1 A Characteristic Experiment in Detail

To study the effects of the proposed beam selection strategies and dominance-based pruning in detail, we focus on a single dataset. For ease of presentation, we choose the (relatively small) *Credit-G* dataset, and we use *WRAcc* as quality measure. We choose a classical Subgroup Discovery setting because it is studied and used by so many people, but this means that we cannot apply compression-based selection. In Figure 1 we have already seen that redundancy is a tremendous problem with DFS top-k subgroup discovery. Hence, we will now apply the proposed beam selection strategies to see if this improves diversity.

Figure 2 shows which subgroups are selected for refinement on each level in the beam search. Clearly, the description-based and cover-based strategies select subgroups from a much wider range than the standard top-100, which is likely to result in a more diverse beam. As expected, a higher degree of redundancy elimination results in more (high-quality but similar) candidates being skipped.

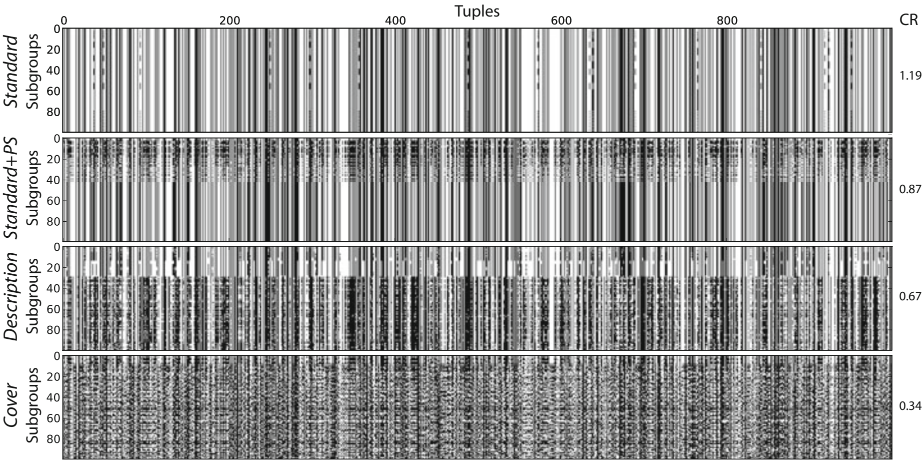


Fig. 3. Subgroup covers obtained with 4 beam search strategies: standard, standard with cover-based post-selection, description-based, and cover-based. Shown are the covers (in *black*) of the top-100 subgroups obtained on *Credit-G* with *WRAcc*. Cover Redundancies (CR) computed from the subgroup sets are shown on the right.

Our hypothesis, of course, is that this more diverse beam selection also results in a more diverse set of results. To assess this, consider the subgroup covers of the 100 subgroups that are obtained after post-selection, in Figure 3. The plots confirm that diversity increases as higher degree redundancy is eliminated: subgroup covers become more and more scattered over all tuples, and CR decreases with each new strategy (from top to bottom). Post-selection seems to perform well at first with *Standard+PS*, but after choosing about 40 subgroups there are no diverse and high-quality candidates left in the remaining 9,960 subgroups, and homogeneity is the end result.

The goal we stated in Section 3 is to find a non-redundant set of high-quality subgroups. It is therefore important that the maximum quality of a subgroup set, the highest quality obtained by any subgroup, does not decrease when using our beam selection strategies.

To assess this, consider the qualities of the 100 subgroups that are obtained after post-selection, in Figure 4. The maximum obtained quality is (almost) the same for all settings, indicating that exploitation does not suffer from beam diversity; a good result. The lower average qualities and larger standard deviations are natural consequences of the diversity enforced by subgroup set selection.

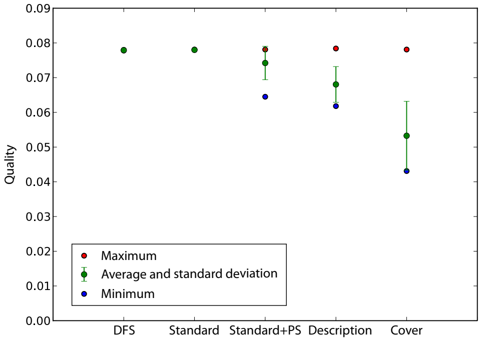


Fig. 4. Qualities of 100 subgroups obtained with different search strategies

6.2 Quantitative Results

We now present results obtained on a large set of experiments, to show that the proposed beam selection strategies have a positive effect in the large majority of cases. That is, resulting subgroup sets are more diverse (and thus less redundant), while not giving in on maximum quality.

For the SD setting, we performed experiments with 3 datasets (*Adult-SD*, *Credit-G* and *Mushroom*), quality measures *WRAcc* and *WKL* and 6 search strategies. These were depth-first search with cover-based post-selection, beam search with a standard beam with and without cover-based post-selection, and beam search with the three proposed selection strategies. The compression-based strategy does not work with *WRAcc*, and DFS with *Adult-SD* and *WKL* was excluded due to a very long runtime (> 2 weeks). Taking this into account, the setup resulted in a total of 26 experiments.

Aggregated results obtained for these experiments are shown in Table 2. A search strategy is better than others if it more often achieves 1) a higher maximum quality, and 2) a lower cover redundancy. This is represented by the average rank results. For each combination of dataset and quality measure, experiments with all search strategies were performed and ranked with respect to 1) maximum quality obtained (φ^{max}, descending), and 2) cover redundancy of the attained subgroup set (CR, ascending). Tied ranks are assigned the average of the ranks for the range they cover. Finally, all ranks for a specific search strategy are averaged.

The results in Table 2 show that DFS with cover-based post-selection needs many candidates and considerable computation time to obtain subgroup sets that are hardly diverse and do not attain the highest maximum quality. The latter is partly due to the restrictions we had to impose on the hypothesis space; multiple conditions on a single attribute (often beneficial) were banned.

The slightly higher average rankings (with respect to maximum quality) of the description-based and cover-based strategies show that diverse beam selection has a modest positive impact on beam search's capability of finding high-quality solutions. A standard beam search with cover-based post-selection gives more diverse results than the description-based strategy, but the latter is faster and it is evidently more diverse than beam search without any post-processing.

Table 2. Subgroup Discovery results, aggregated over 3 datasets and 2 quality measures. Shown are the average number of candidates, time per experiment, subgroup description sizes (#conditions), subgroup sizes and cover redundancies. On the right, average ranks are given as obtained by ranking experiments stratified by strategy.

Search strategy	*Experiment avg*		*Subgroup set avg*			*Rank avg*	
	$\|Cands\|$	time (min)	descr.	size	CR	φ^{max}	CR
DFS + PS	403801872	1553	3.5	5712	1.10	3.4	3.8
Standard	88641	0.3	4.7	6535	1.23	3.2	4.0
Standard + PS	88641	4.2	3.6	7494	0.80	3.2	2.5
Description	88508	1.0	4.3	6591	0.98	2.8	3.7
Cover	89116	49	4.4	8758	0.37	2.8	1.0
Compression	87304	16	2.7	3296	1.12	3.3	3.0

Table 3. Exceptional Model Mining results, aggregated over 4 datasets and 2 quality measures. Shown are the average number of candidates, time per experiment, subgroup description sizes (#conditions), subgroup sizes and cover redundancies. On the right, average ranks are given as obtained by ranking experiments stratified by strategy.

Search strategy	*Experiment avg*		*Subgroup set avg*			*Rank avg*	
	$\lvert Cands \rvert$	time (min)	descr.	size	CR	φ^{max}	CR
Standard	244830	8	4.8	4840	1.53	3.1	4.6
Standard + PS	244830	52	3.4	5397	1.07	2.6	2.5
Description	244659	49	3.8	5163	1.36	1.9	3.5
Cover	244830	62	3.4	5493	0.48	3.2	1.2
Compression	255992	143	2.1	653	1.07	3.8	2.4

When the cover-based strategy is incorporated *within* the search, however, the results stand out with respect to cover diversity. The downside is that it needs more time, but it is still very fast when compared to DFS. Compression-based selection does not seem to work well in the SD setting, which is not unexpected since only a very limited number of distributions can be distinguished with a single binary model attribute.

We performed EMM experiments on 4 datasets (*Adult-EMM*, *Emotions*, *Mammals*, and *Yeast*), with quality measures *WKL* and *WKG* and 5 beam search strategies. *WKG* was not used in combination with *Mammals*, since the induction of Krimp code tables takes too long on this dataset; *WKL* is a good and fast alternative. We chose to apply the combination of *WKG* and compression-based selection only to *Emotions*, as all models can be cached in memory for this dataset. The results of the 26 experiments are presented in Table 3.

The results for EMM are slightly different from those for SD. Description-based selection finds better overall solutions than the other strategies. It performs better than *Standard* in terms of cover redundancy, but not better than the 2-step *Standard+PS* approach. For fast mining of high-quality results, the description-based strategy seems a good choice. Dominance-based pruning is not applied with *Standard*, resulting in lower maximum qualities than with *Standard+PS*.

As expected from its basic principle, cover-based selection is again the clear winner with respect to cover diversity: it achieves the lowest cover redundancies. The compression-based scheme gives slightly lower maximum qualities, but the subgroups are quite diverse, smaller and have shorter descriptions.

We performed a Friedman test on 8 rankings obtained with the compression-based quality measures, to be able to include the compression-based strategy in the comparison. For each of the 7 datasets, a ranking was obtained with *WKL*, 1 ranking came from *Emotions* with *WKG*. Between the φ^{max} rankings, no significant differences were found; the 5 strategies exhibit no significant differences with respect to exploitation. In the CR rankings, significant differences were found (p-value = 0.00004), and we did a post-hoc Wilcoxon-Nemenyi-McDonald-Thompson test. *Standard+PS*, *Cover* and *Compression* have significantly better rankings than *Standard*, and *Cover* is also significantly better than *Description*.

All in all, incorporating subgroup selection *within* beam search yields clearly better results than applying it as post-processing step. Employing the description-based selection scheme comes at little computational cost, but does give higher-quality and more diverse results than without using any subgroup selection techniques. At the expense of some more computation time, cover-based selection eliminates more redundancy and results in a much more diverse subgroup set. The compression-based method does not always work well, but should be employed for datasets where many underlying distributions are present in the model data, such as it is the case for e.g. *Mammals*.

Finally, we consider the effect of dominance-based pruning on the subgroup sets. In the SD experiments, the average number of conditions per subgroup description decreases from 4.5 to 3.4 and average subgroup quality increases with 4% on average. For EMM, the effect is even larger and the average number of conditions decreases from 4.9 to 3.0, an average decrease of 1.9 conditions per description! Meanwhile, average subgroup quality increases with 20.3% on average. Note that these changes are due to both the pruning of individual descriptions and the removal of syntactically identical subgroups.

7 Related Work

To the best of our knowledge, we are the first to combine pattern selection techniques and beam search to achieve non-redundant Generalised Subgroup Discovery. Kocev et al. [8] previously proposed to incorporate similarity constraints in a beam search to improve the induction of predictive clustering trees.

Several methods have been proposed to address the redundancy problem in SD/EMM. Garriga et al. [2] proposed closed sets for labeled data, but similar to closed frequent itemsets, this only eliminates a limited part of redundancy as only individual patterns are considered. An advantage is that 'relevant' subgroups can be efficiently mined [12]. A downside is that it does not apply to the EMM setting. We previously proposed the EMDM algorithm [10], but this method does not apply to the SD setting and for the EMM setting, it is dependent on the initial candidates and it finds more complex subgroup descriptions.

The beam selection strategies we propose are clearly inspired by pattern set selection methods such as those proposed by Bringmann & Zimmerman [1] and Peng et al. [15]. The key difference is that we perform pattern selection *within* a discovery algorithm to improve the end result.

8 Conclusions

Effective and efficient heuristics are crucial for performing discovery tasks in large and complex data. In addition to that, the incredible amount of redundancy in hypothesis spaces renders straightforward top-k mining useless. We address these problems by incorporating heuristic pattern set selection methods *within* a beam search, thereby improving the balance between exploration and exploitation.

We described three degrees of redundancy and introduced a subgroup set selection strategy for each degree. Experiments with both Subgroup Discovery and Exceptional Model Mining show that the proposed methods for *subgroup set mining* return high-quality yet diverse results. The three methods offer the data miner a trade-off between redundancy elimination and computation time.

Acknowledgments. This research is financially supported by the Netherlands Organisation for Scientific Research (NWO) under project number 612.065.822.

References

1. Bringmann, B., Zimmermann, A.: The chosen few: On identifying valuable patterns. In: Proceedings of the ICDM 2007, pp. 63–72 (2007)
2. Garriga, G.C., Kralj, P., Lavrac, N.: Closed sets for labeled data. Journal of Machine Learning Research 9, 559–580 (2008)
3. Grosskreutz, H., Rüping, S., Wrobel, S.: Tight optimistic estimates for fast subgroup discovery. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 440–456. Springer, Heidelberg (2008)
4. Grünwald, P.D.: The Minimum Description Length Principle. MIT Press, Cambridge (2007)
5. Heikinheimo, H., Fortelius, M., Eronen, J., Mannila, H.: Biogeography of european land mammals shows environmentally distinct and spatially coherent clusters. J. Biogeography 34(6), 1053–1064 (2007)
6. Klösgen, W.: Explora: A Multipattern and Multistrategy Discovery Assistant. In: Advances in Knowledge Discovery and Data Mining, pp. 249–271 (1996)
7. Knobbe, A., Ho, E.K.Y.: Pattern teams. In: Proceedings of the ECML PKDD 2006, pp. 577–584 (2006)
8. Kocev, D., Struyf, J., Džeroski, S.: Beam search induction and similarity constraints for predictive clustering trees. In: Džeroski, S., Struyf, J. (eds.) KDID 2006. LNCS, vol. 4747, pp. 134–151. Springer, Heidelberg (2007)
9. Lavrač, N., Kavšek, B., Flach, P., Todorovski, L.: Subgroup discovery with cn2-sd. J. Mach. Learn. Res. 5, 153–188 (2004)
10. van Leeuwen, M.: Maximal exceptions with minimal descriptions. Data Min. Knowl. Discov. 21(2), 259–276 (2010)
11. Leman, D., Feelders, A., Knobbe, A.: Exceptional model mining. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 1–16. Springer, Heidelberg (2008)
12. Lemmerich, F., Rohlfs, M., Atzmüller, M.: Fast discovery of relevant subgroup patterns. In: Proceedings of FLAIRS (2010)
13. Mannila, H., Toivonen, H.: Multiple uses of frequent sets and condensed representations. In: Proceedings of the KDD 1996, pp. 189–194 (1996)
14. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 398–416. Springer, Heidelberg (1998)
15. 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 27(8), 1226–1238 (2005)
16. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Mining and Knowledge Discovery 23(1), 169–214 (2011)
17. Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: Komorowski, J., Żytkow, J.M. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 78–87. Springer, Heidelberg (1997)

Larger Residuals, Less Work: Active Document Scheduling for Latent Dirichlet Allocation

Mirwaes Wahabzada* and Kristian Kersting*

Knowledge Discovery Department, Fraunhofer IAIS
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
firstname.lastname@iais.fraunhofer.de

Abstract. Recently, there have been considerable advances in fast inference for latent Dirichlet allocation (LDA). In particular, stochastic optimization of the variational Bayes (VB) objective function with a natural gradient step was proved to converge and able to process massive document collections. To reduce noise in the gradient estimation, it considers multiple documents chosen uniformly at random. While it is widely recognized that the scheduling of documents in stochastic optimization may have significant consequences, this issue remains largely unexplored. In this work, we address this issue. Specifically, we propose residual LDA, a novel, easy-to-implement, LDA approach that schedules documents in an informed way. Intuitively, in each iteration, residual LDA actively selects documents that exert a disproportionately large influence on the current residual to compute the next update. On several real-world datasets, including 3M articles from Wikipedia, we demonstrate that residual LDA can handily analyze massive document collections and find topic models as good or better than those found with batch VB and randomly scheduled VB, and significantly faster.

1 Introduction

Latent Dirichlet allocation (LDA) has recently become very popular due to its effectiveness at extracting low-dimensional representations from sparse high-dimensional data, with numerous applications in areas such as text analysis and computer vision [1]. Unfortunately, fitting a LDA topic model given a set of training documents requires approximate inference techniques that are computationally expensive. This makes it challenging to apply LDA to large-scale document collections that nowadays become increasingly common. Such datasets originate for example from online books at Google or image collections at Flickr. And as storage capacity continues to expand, today's "large" is certainly tomorrow's "medium" and next week's "small". For instance International Data Corporation[1] (IDC), a consultancy, has estimated that in August 2010 the amount of information available on the Internet did surpass the barrier of $1\text{ZB} = 10^{21}\text{B}$.

[*] Both authors contributed equally.

[1] Mar 2011, http://www.idc.com/getdoc.jsp?containerId=prUS22723811

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 475–490, 2011.

According to Eric Schmidt, CEO of Google, this amount currently grows at a rate of $2.5\text{EB} = 2.5 \cdot 10^{18}\text{B}$ per day. Most of these data consist of user generated content such as videos, photos, blogs, or reviews. Apparently, processing such large-scale document collections opens up completely new and interesting applications for machine learning techniques in general and LDA in particular.

A promising approach to scaling LDA to large data sets are online variants, see e.g. [23,2,11,16] and references in there, that incrementally build topic models when a new document (or a set of documents) appears. For instance, Hoffman *et al.* [11] presented an online variational Bayes (VB) algorithm for LDA based on online stochastic optimization with a natural gradient step that can easily analyze massive document collections. In addition to providing a solution to the problem of growing document collections, online algorithms also open up different avenues for parallelization of inference from batch algorithms, providing ways to draw on the enhanced computing power of multiprocessor systems, and different tradeoffs in runtime and performance from other algorithms; another common approach to scaling LDA, see e.g. [18,15,22] and reference in there. Here, we explore another avenue opened up by online LDA algorithms, namely, to revisit batch LDA and ask the question whether we can improve it by viewing it as a quasi-online approach that processes documents respectively mini-batches one at a time? Somewhat surprisingly, there has been virtually no attempt to study the question of determining a good order for documents to be processed. While it is widely recognized that the scheduling of documents in stochastic optimization of LDA topic models may have significant consequences, this issue remains largely unexplored. Instead, practitioners schedule documents essentially uniformly at random, perhaps due to ease of implementation, and to the lack of clear guidelines on scheduling the documents. In this work, we address the question of how to schedule documents and show that convergence can be reached faster.

Specifically, triggered by recent results on randomized low-rank matrix factorization approaches [7,5,19,14] — they essentially randomly sample columns, i.e., documents according to a probability distribution that depends on the Euclidean norms of those documents — it was recently proposed to schedule documents that exert a disproportionately large influence on the topics of the corpus before less influential documents. Naively instantiating this idea results in a fix schedule: sort documents randomly biased towards those ones with higher norms. Then, we simply run online LDA following this fix schedule. In fact, this influence scheduled LDA (isLDA) can already result in considerable efficiency gains. However, as we will show in this paper, we can do considerably better. While isLDA is indeed a valid approach, it suffers from one drawback. Since it starts optimizing long before having seen the entire document collection even once, it tends to overfit on documents with many words; they are seen much earlier than documents with few words. Consequently, isLDA has to repair for this bias by several passes over the entire dataset. We therefore propose a novel schedule that overcomes this drawback. Intuitively, we schedule documents dynamically in each pass over the entire document collection according to a probability dis-

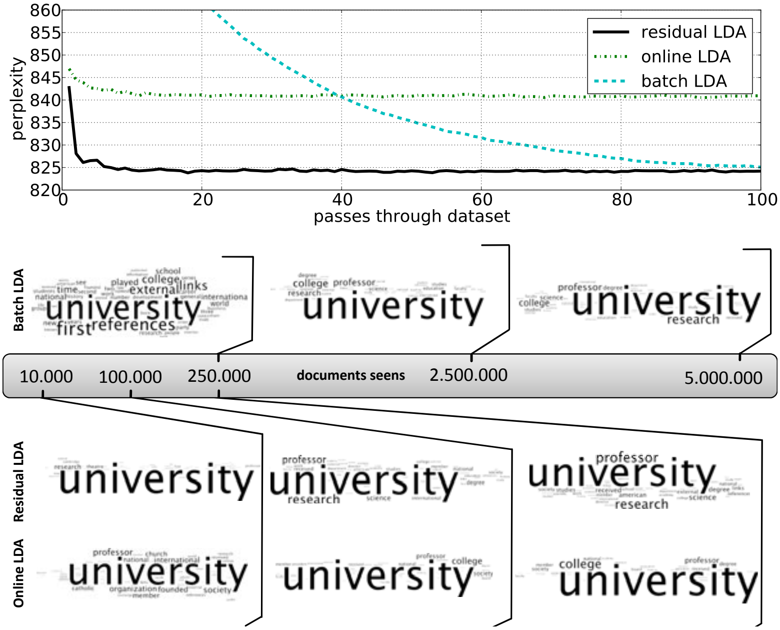


Fig. 1. Residual LDA can handily analyze massive document collections and finds topics better than those found with batch LDA and online LDA, and significantly faster: (Top) Held-out perplexity (the lower, the better) results on 3M Wikipedia articles as a function of full passes through the dataset. (Bottom) Evolution of the topic "university research" (bottom) after having seen 10K, 100K, and 250K (residual and online LDA) respectively 250K, 2.5m, 5m articles (batch LDA) on 250K Wikipedia. The 50 most likely words are shown as word cloud. The word size is proportional to the probability. Residual LDA is qualitatively more similar to batch LDA than online LDA is, university, professor and research prominently appear in both of them. This is also supported by smaller Hellinger distance. (Best viewed in color)

tribution that depends on the Euclidean norms of the current residuals of the documents. As we will show, this dynamic schedule can actually be turned into an active schedule: in each iteration we only consider a small, informative subset of the documents sampled according to the residual distribution. This is sensible because, if we overfit on the current active set of documents, the residuals of the currently inactive documents will get larger. Moreover, it establishes a novel link between LDA and active learning, see [17] for an overview. Whereas our active schedule is unsupervised, most active learning approaches make use of label information. Even recent active clustering approaches, see e.g. [21], typically provide semi-supervision in terms of must-link and cannot-link constraints. Closest in spirit is probably early work on active data sampling [13]. Actually, as we will point out in the conclusions, there is also a close connection to search distributions for optimization [24].

We evaluated the resulting LDA, called *residual LDA*, both qualitatively and quantitatively using several benchmark datasets, including 3M articles from Wikipedia. The experimental results demonstrate that — as can also be seen in Fig. 1 — residual LDA can handily analyze massive document collections and find topic models as good or better than those found with randomly and informatively scheduled LDA, and significantly faster. In other words "larger residuals, less work", which also explains the title of the paper.

We proceed as follows. After reviewing LDA based on variational Bayes, we motivate residual LDA by drawing connections between LDA and low-rank matrix factorization. Based on this connection, we then introduce residual LDA and devise a active schedule that prefers documents with high influence on the gradient over less influence ones. Before concluding, we present our experimental evaluation.

2 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is a Bayesian probabilistic model of collections of text documents [1]. It assumes a fixed number of K underlying topics in a document collection D. Topics are assumed to be drawn from a Dirichlet distribution, $\beta_k \sim Dir(\eta)$, which is a convenient conjugate to the multinomial distribution of words appearing in documents. According to LDA, documents are generated by first drawing topic proportions according to $\theta_d \sim Dir(\alpha)$, where α is the parameter of the Dirichlet prior on the per-document topic distributions. Then for each word i a topic is chosen according to $z_{di} \sim Mult(\theta_d)$ and the observed word w_{di} is drawn from the selected topic, $w_{di} \sim Mult(\beta_{z_{di}})$. We assume symmetric priors for θ and β but asymmetric ones are possible, see e.g. [20]. The true posterior distribution can not be computed directly and is usually approximated using Markov chain monte carlo (MCMC) or variational inference. In this paper, we focus on variational Bayesian (VB) inference. Here, the true posterior is approximated using a simpler, fully factorized distribution q. Following Blei *et al.* and Hoffman *et al.* [1,11], we choose $q(z, \theta, \beta)$ of the form $q(z_{di} = k) = \phi_{dw_{di}k}$, $q(\theta_d) = Dir(\theta_d, \gamma_d)$, and $q(\beta_k) = Dir(\beta_k, \lambda_k)$. The posterior over the per-word topic assignments z is parameterized by ϕ, the posterior over the per-document topic weights θ is parameterized by γ, and the posterior over the topics β is parameterized by λ. These variational parameters are optimized to maximize the Evidence Lower BOund (ELBO) $\log p(w \mid \alpha, \eta) \geq \mathcal{L}(w, \phi, \gamma, \lambda) \triangleq$

$$\mathbb{E}_q \left[\log p(w, z, \theta, \beta \mid \alpha, \eta)\right] - \mathbb{E}_q \left[\log q(z, \theta, \beta)\right], \tag{1}$$

which is equivalent to minimizing the Kullback-Leiber divergence between $q(z, \theta, \beta)$ and the true posterior $p(z, \theta, \beta \mid w, \alpha, \eta)$. It can be shown that Eq. (1) factorizes to a summation over documents d:

$$\mathcal{L}(w, \phi, \gamma, \lambda) = \sum\nolimits_d \Big\{ \mathbb{E}_q \left[\log p(w_d \mid \theta_d, z_d, \beta)\right] + \mathbb{E}_q \left[\log p(z_d \mid \theta_d)\right] - \mathbb{E}_q \left[\log q(z_d)\right]$$
$$+ \mathbb{E}_q \left[\log p(\theta_d \mid \alpha)\right] - \mathbb{E}_q \left[\log q(\theta_d)\right] + \left(\mathbb{E}_q \left[\log p(\beta \mid \eta)\right] - \mathbb{E}_q \left[\log q(\beta)\right]\right) / \mathbf{D} \Big\}.$$

Algorithm 1. A single iteration of Variatonal Bayes for LDA (vbLDA). Setting $\rho_t = 1$ and $D = \tilde{D}$ corresponds to running batch LDA

```
Input: $\tilde{D}$, $\rho_t$
Output: $\boldsymbol{\lambda}$,$\boldsymbol{\gamma}_{\tilde{D}}$
foreach document $d$ in $\tilde{D}$ do
    Initialize $\gamma_{dk} = 1$;
    /* The const. is arbitrary                                         */
    repeat
        Set $\phi_{dwk} \propto \exp\{\mathbb{E}_q[\log\theta_{dk}] + \mathbb{E}_q[\log\beta_{kw}]\}$;
        Set $\gamma_{dk} = \alpha + \sum_w \phi_{swk} n_{dw}$;
    until $\frac{1}{K}\sum_k |change\ in\ \gamma_{dk}| < 0.00001$ ;
/* Compute M step                                                      */
Compute $\tilde{\lambda}_{kw} = \eta + \frac{\mathbf{D}}{|\tilde{D}|}\sum_{d \in \tilde{D}} n_{dw}\phi_{dwk}$;
Set $\boldsymbol{\lambda} = (1-\rho_t)\boldsymbol{\lambda} + \rho_t\tilde{\boldsymbol{\lambda}}$;
```

When using VB (as opposed to MCMC) documents can be summarized by their word counts, i.e., $\mathcal{L}(w,\phi,\gamma,\lambda) \triangleq \sum_{d=1}^{\mathbf{D}} \ell(n_d,\phi_d,\gamma_d,\lambda)$, where $\mathbf{D}$ denotes the number of documents, n_d the word count vector and $\ell(n_d,\phi_d,\gamma_d,\lambda)$ the contribution of document d to the ELBO. Now, $\mathcal{L}$ can be optimized using coordinate ascent over the variational parameters ϕ, γ, λ (see [1,11] for more details),

$$\phi_{dwk} \propto \exp\{\mathbb{E}_q[\log\theta_{dk}] + \mathbb{E}_q[\log\beta_{kw}]\},$$
$$\gamma_{dk} = \alpha + \sum\nolimits_w n_{dw}\phi_{dwk}, \text{ and } \lambda_{kw} = \eta + \sum\nolimits_d n_{dw}\phi_{dwk}, \tag{2}$$

iteratively optimizing each variational parameter to increase the objective. The conditional expectations under q of $\log\theta$ and $\log\beta$ are

$$\mathbb{E}_q[\log\theta_{dk}] = \Psi(\gamma_{dk}) - \Psi\left(\sum_{i=1}^{K}\gamma_{ki}\right) \text{ and } \mathbb{E}_q[\log\beta_{kw}] = \Psi(\lambda_{kw}) - \Psi\left(\sum_{i=1}^{W}\lambda_{di}\right),$$

where W is the size of the vocabulary, and Ψ denotes the digamma function. The updates in (2) are guaranteed to converge to a stationary point of the ELBO. By analogy to the EM algorithm, we can partition these updates into an *E-step* - iteratively updating γ and ϕ until convergence, holding λ fixed - and an *M-step* - updating λ given ϕ. In practice, this algorithm converges to a better solution if we reinitialize γ and ϕ before each *E-step*.

Based on VB, Hoffman *et al.* [11] have introduced an online variant that we here present for the batch case running over mini-batches (chunks of multiple observations) as summarized in Alg. 2. That is, we assume that the corpus of documents has been sorted according to some schedule, i.e. permutation $\pi(i)$ and chunked into l mini-batches $B_1, B_2, \ldots, B_l$ of size S with $B_i = d_{(i-1)\cdot S+1}, d_{(i-1)\cdot S+2}, \ldots, d_{i\cdot S}\}$ (w.l.o.g, we assume $n = l \cdot S$). That is, the ELBO $\mathcal{L}$ is set to maximize $\mathcal{L}(w,\phi,\gamma,\lambda) \triangleq \sum_{B_i}\sum_{d \in B_i} \ell(n_d, \phi_d(n_d,\lambda), \gamma_d(n_d,\lambda), \lambda)$. For each mini-batch, we perform an *E step* to find locally optimal values of γ_t

Algorithm 2. Online LDA (oLDA)

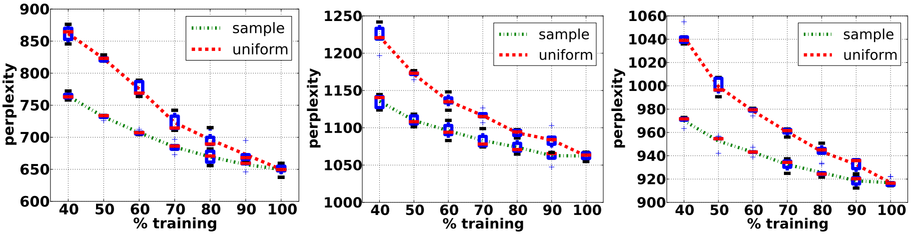
```
Input: $D$ (documents), $S$ (batchsize)
Define $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$ with $\kappa \in (0.5, 1]$;
Initialize $\boldsymbol{\lambda}$ randomly and set $t = 0$;
repeat
    Select $S$ documents randomly forming the mini-batch $\tilde{D}$;
    Compute $(\boldsymbol{\lambda}, \boldsymbol{\gamma}_{\tilde{D}}) = \mathrm{vbLDA}(\tilde{D}, \boldsymbol{\lambda}, \rho_t)$ using Alg. 1;
    Increment $t := t + 1$;
until converged ;
```

and ϕ_t, holding λ fixed. In the following *M step*, we compute $\tilde{\lambda}$, the setting of λ that would be optimal (given ϕ_t) if the entire corpus consisted only of repetitions of the batch B_i. λ is updated through a weighted average of its previous value and $\tilde{\lambda}$ computed in the current *M step*. The rate of change ρ_t is set to $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$. Here, $\tau_0 \geq 0$ slows down early iterations of the algorithm, and $\kappa \in (0.5, 1]$ controls the influence of old values of $\tilde{\lambda}$ and ensures convergence.

As Hoffman *et al.* [11] have shown this mini-batch VB-LDA corresponds to a stochastic natural gradient algorithm on the variational objective $\mathcal{L}$. Using mini-batches reduces the noise in the stochastic gradient estimation as we consider multiple observations per update, i.e., $\tilde{\lambda}_{kw} = \eta + \mathbf{D}/S \sum_{s \in B_i} n_{sw} \phi_{skw}$ where n_{sw} is the s-th document in the i-th mini-batch. The variational parameters ϕ_{ts} and γ_{ts} for this document are fit with a normal E step. Note that we recover batch VB when we set $\pi(i) = i$, the batch size to $S = \mathbf{D}$, and $\kappa = 0$. The benefits of this LDA approach are manifold, see [11]. It empirically converges faster than batch collapsed Gibbs sampling. It does not require a full pass through the entire corpus in order to compute an update. Instead, it makes an update per mini-batch and in turn can be quite fast when applying to large datasets. Furthermore, it converges as long as the expected number of times we see each document is the same for each document.

3 isLDA ≈ Random Matrix Factorization

It is known that LDA and its precursor probabilistic latent semantic analysis (pLSA) [12] are closely related. In particular, one can show that pLSA is tantamount to LDA with a uniform prior [10]. Given this connection, one can also establish a relation between LDA and certain settings of low-rank matrix factorization. Specifically, Gaussier and Goutte [8] and Ding *et al.* [4] have noted that pLSA correspond to specific instances of the problem of non-negative matrix factorization. pLSA can thus be reduced to a low-rank matrix factorization problem. Consequently, LDA topic models can be determined using algorithmic approaches that differ from the conventional idea of expectation maximization, MCMC, and VB. In particular, randomized matrix factorization approaches become applicable to LDA as well. For instance, Frieze *et al.* [7] introduced a sampling approach, in which the rows of a matrix are picked with probabilities proportional to their squared lengths. This and similar randomized matrix



(a) **WebKB** (b) **N20** (c) **Wiki 25K**

Fig. 2. More influence, lower perplexity; lowest perplexity when using the full training set: Held-out perplexity results (the lower, the better) for different proportions of the original dataset. We selected $40\%, 50\%, \ldots, 100\%$ documents and ran LDA for $K = 100$. Results are averaged over five reruns. (Best viewed in color)

factorization approaches have been proved to approximately minimize the reconstruction error in terms of the Frobenius norm and be successful in several tasks and applications, see also e.g [5,19,14].

Given the link between low-rank matrix factorization and LDA, it is natural to ask "Can we improve LDA by adapting techniques developed for matrix factorization?" Recently, we and colleagues have shown that this is indeed the case. A common randomized matrix factorization approach, see e.g. [5,19], is to approximate a given matrix A by S rescaled rows/columns sampled from A. To do so, we compute an importance score for each row, and sample rows using that score as an importance sampling probability distribution. A common score for matrix factorization is

$$p(i) = \sum\nolimits_j n_{ij}^2 / \sum\nolimits_{i,j} n_{ij}^2 \ , \tag{3}$$

and the rescaling factor is $1/\sqrt{p(i) \cdot S}$. Thus, this importance score depends on the whole corpus and intuitively captures the "influence" of a given document on the LDA topic model. By preferentially choosing documents that exert a disproportionately large influence on the topic model, we expect to capture the important part of a given corpus at hand. Fig. 2 illustrates that this is actually the case. Using the probability (3) as importance score yields significantly lower perplexities than selecting documents uniformly at random when selecting only $40\%, 50\%, \ldots, 100\%$ documents of the original corpus. In other words, the way we schedule documents (and in turn the way we build mini-batches) can make a crucial difference on how well and long mini-batch LDA takes to converge.

Based on this insight, we can devise a naive "fix" schedule for LDA that is easy-to-implement. Essentially, we apply the importance sampling procedure to LDA. However, whereas the randomized matrix factorization approaches sample a subset of documents with replacement, we keep all documents exactly once. Consequently, we compute a schedule by sampling all documents without replacement biased towards those ones with higher norms. In other words, the documents with higher importance score will have higher chance to be processed

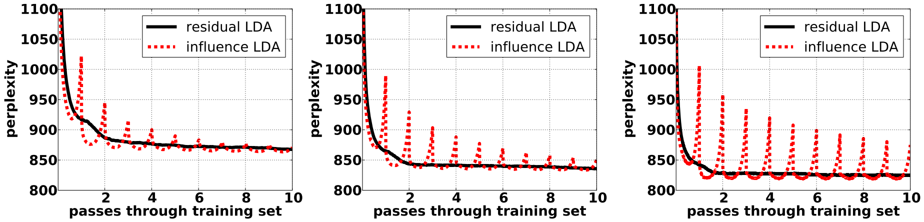


Fig. 3. Residual LDA avoids overfitting on large documents. Held-out perplexity results (for $K = 100$) as a function of passes through training set on Wikipedia (from left to right 50K, 250K, and 3M documents). The dashed line denotes the performance of isLDA, the solid lines that of residual LDA. (Best viewed in color)

earlier. We and colleaguescalled the resulting LDA — compute $p(i)$ as in Eq. (3), rescale and shuffle the documents according to π, build mini-batches as described earlier, and run Alg. 1 — *influence scheduled* LDA (isLDA).

However, while isLDA is indeed a valid approach and provably converges, it suffers from one major drawback. Since isLDA starts optimizing long before having seen the entire document collection even once, it tends to overfit on documents with many words. They are seen much earlier than the documents with few words only. Consequently, isLDA has to repair for this bias by several passes over the entire dataset. This is illustrated in Fig. 3. As one can see, the held-out perplexity drops after each full pass over the entire document collection. During one pass over the entire document collection, however, influence LDA overfits on the large documents. One is tempted to simply use the first half of the data set only since the minimal held-out perplexity per pass is achieved at this point. This, however, is not sensible. Recall the results presented in Fig. 2. They illustrate that omitting least important documents actually decreases the overall performance. Only when the entire document collection is potentially considered for training, the best overall performance is achieved.

In other words, isLDA is simply too static. The fix schedule does not allow to change the order of documents processed during learning, although the impact of a document on the LDA topic model is indeed constantly changing. A dynamic schedule would be more sensible. How do we realize this? Intuitively, we schedule documents in each pass over the entire document collection according to a probability distribution that depends on the Euclidean norms of the residuals of those documents. This is not only sensible but actually can also be turned into an active schedule: in each iteration we only consider a small, informative subset of the documents sampled according to the residual distribution. Indeed, we may overfit on the current active documents. However, their importance score will drop, and the residuals of the currently inactive documents will increase. That is, their importance score increases and so does the probability of selecting an inactive document and making it active. This is the basic idea underlying residual LDA that we will now introduce.

4 Residual Latent Dirichlet Allocation

Intuitively, we want to select documents which are most informative with respect to the variational bound. So, what is the impact of a single document on the current variational bound? How do we compute this per document influence?

Recall that VB uses coordinate ascent to maximize the objective function. That is, each variational parameter $\boldsymbol{\gamma}$, $\boldsymbol{\phi}$, respectively $\boldsymbol{\lambda}$ is varied while all others are held constant. Since we are only interested in the influence of a single document d on the variational bound, however, it is sufficient to consider $\boldsymbol{\phi}$ only. That is, we view the "per document d influence" as the change of the variational parameter $\boldsymbol{\phi}$ for each word w in d $\tilde{\nabla}_d^t := \sum_k \sum_w n_{dw} \mid \phi_{dwk}^t - \phi_{dwk}^{t-1} \mid$. Here, ϕ_{wk}^t is the optimized parameter at iteration t representing the probability that w is generated by topic k, and n_{dw} represents the number of times w appears in d.

This importance measure essentially needs $\mathcal{O}(U \cdot K \cdot \mathbf{D})$ storage effort with $U = \max_d |w_d|$. We have to store the values $\boldsymbol{\phi}$ of all unique words in document d times the number of topics K. This is essentially intractable for large document collections. To restrict the overall complexity, we therefore summarize $\boldsymbol{\phi}$ for each topic in a document. More formally, using the partial derivative of the Eq. (1) w.r.t. γ_{dk} it is shown that it yields a maximum at $\gamma_{dk} = \alpha + \sum_w n_{dw}\phi_{dwk}$ (see e.g. [1]). Additionally applying the triangle inequality, we arrive at

$$\tilde{\nabla}_d^t = \sum_k \sum_w \mid n_{dw}\phi_{dwk}^t - n_{dw}\phi_{dwk}^{t-1} \mid \geq \sum_k \mid \sum_w n_{dw}\phi_{dwk}^t - \sum_w n_{dw}\phi_{dwk}^{t-1} \mid$$

$$= \sum_k \mid \alpha + \sum_w n_{dw}\phi_{dwk}^t - (\alpha + \sum_w n_{dw}\phi_{dwk}^{t-1}) \mid = \sum_k \mid \gamma_{dk}^t - \gamma_{dk}^{t-1} \mid =: \nabla_d^t \, . \quad (4)$$

This measure is intuitive for finding high impact documents, since the documents that have the highest ∇_d^t are the ones we are most uncertain about. Unfortunately, it does not consider the expected reduction or growth in uncertainty. Therefore, we instead search for highly informative documents by maximizing the information gained about ∇_d^t. More formally, the influence $\xi(d)$ of a document d is $\xi(d) = \max\{c \, , \nabla_d^t - \nabla_d^{t-1}\}$. Here, $c > 0 \in \mathbb{R}$ is some small constant that prevents the influence to become too small or even zero. Now, following Eq. 3, the importance score $\mathbf{p}(d)$ of the document d is: $\mathbf{p}(d) = \xi(d)^2 / \sum_i \xi(i)^2$ Proceeding as for LDA — run Alg. 1 computing $p(i)$ using this new influence score, rescaling, and shuffling the documents according to π in each pass over the entire document collection — essentially yields *residual LDA*.

However, we can do better. The dynamic schedule can actually be turned into an active schedule: in each iteration we only consider a small, informative subset of the documents sampled according to the residual distribution. This is sensible because, if we overfit on the current active set of documents, the residuals of the currently inactive documents will get larger. In turn, the probability that they will be selected and become active in the next iteration increases. However, how should we initialize respectively update the values $\boldsymbol{\xi}$ for previously unseen documents? Assuming that the expected variational parameters are uniform for

Algorithm 3. Residual LDA

```
   Input: $D$ (documents), $S$ (batchsize)
 1 Define $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$ with $\kappa \in (0.5, 1]$;
 2 Initialize $\boldsymbol{\lambda}$ randomly, set $t := 0$ and $\sigma = 0$;
 3 foreach $d \in D$ do
 4     $\xi^2(d) := (\sum_w n_{dw})^2$;
 5     $\sigma := \sigma + \xi^2(d)$;
 6 Set $c = \min\{\xi(d)\}$;
 7 repeat
       /* Update document importance scores */
 8     foreach $d \in D$ do
 9         $\mathbf{p}(d) := \sigma^{-1} \cdot \xi^2(d)$;
10     Sample $S$ active documents $\tilde{D}$ according to $\mathbf{p}$;
       /* Run one iteration of variational inference (Alg. 1) */
11     $(\boldsymbol{\lambda}, \boldsymbol{\gamma}_{\tilde{D}}) := \text{vbLDA}(\tilde{D}, \boldsymbol{\lambda}, \rho_t)$;
       /* Update document influence for the active documents in $\tilde{D}$ */
12     foreach $\tilde{d} \in \tilde{D}$ do
13         $\nabla^t_{\tilde{d}} := \sum_k | \gamma^t_{\tilde{d}k} - \gamma^{t-1}_{\tilde{d}k} |$;
14         $\sigma := \sigma - \xi^2(\tilde{d})$;
15         $\xi^2(\tilde{d}) := (\max(c, \nabla^t_{\tilde{d}} - \nabla^{t-1}_{\tilde{d}}))^2$;
16         $\sigma := \sigma + \xi^2(\tilde{d})$;
17     Increment $t := t + 1$;
18 until converged ;
```

documents not seen yet, i.e., $\phi_{dwk} = K^{-1}$, Eq. (4) simplifies as follows: $\xi(d)_{init} =$

$$\sum\nolimits_k | \gamma_{dk} - 0 | - 0 = \sum\nolimits_k \gamma_{dk} = \sum\nolimits_k (\alpha + \sum\nolimits_w n_{dw}\phi_{dwk}) = K\alpha + \sum\nolimits_w n_{dw} \ .$$

Thus, $\xi(d)_{init}$ can essentially be set to the number of words in d, i.e. $\sum_w n_{dw}$.

The overall approach is summarized in Alg. 3. We start off by initializing the influence scores of all documents (lines 3-5). Then, in each iteration, we compute $\mathbf{p}(d)$ (lines 8-9), sample S documents according to $\mathbf{p}(d)$ (line 10), run one iteration of variational inference to the active mini-batch (line 11), update the influence scores of the active documents (line 12-16), and iterate until convergence. For efficiency reasons, we can switch to online LDA in later iterations, i.e., we avoid computing any schedule. We can do so because the gradients will eventually get very similar. When this happens, residual LDA essentially mimics online LDA. To decide when to switch, we can simply check whether the entropy of importance scores drops below some threshold ϵ. If $\epsilon \geq | \sum_d \mathbf{p}(d) \log \mathbf{p}(d) - \log \frac{1}{\mathbf{D}} |$, we switch to online LDA.

5 Experimental Evaluation

Our intention here is to investigate the following questions: **(Q1)** Can residual LDA be faster than batch LDA, online LDA and isLDA? **(Q2)** If so, does residual

LDA find solutions that are as good as batch LDA? **(Q3)** Does residual LDA scale better than online LDA and isLDA in terms of number of documents resp. the size of the vocabulary?

To this aim, we implemented residual LDA, batch LDA and isLDA in Python based on Hoffman *et al.*'s [11] Python code[2] for online LDA. We evaluated the performance of the methods on several datasets where **D** denotes the number of documents, **W** the number of unique words, and **N** the number of terms. The **WebKB** dataset consists of webpages of various universities with four different categories (student, course, faculty, project). We used the dataset provided by Ana Cardoso-Cachopo[3] with **D** = 3869 and **N** = 217671. We chose a vocabulary of **W** = 3000 unique words consisting of the terms with the highest TFIDF (term frequency inverse document frequency). The Reuters dataset **R8**, a classic corpus for text classification algorithms, contains newswire articles. We used the version provided by Ana Cardoso-Cachopo[3] with **D** = 5378 and **N** = 234650. As for **WebKB**, we chose a vocabulary of **W** = 3000 unique words consisting of the terms with the highest TFIDF. Finally, we also used the 20-newsgroups dataset **20N** as in [9][4] with **D** = 18576, **N** = 1847456, and **W** = 10000. From all corpora, we removed documents with less than six words.

For evaluation we computed perplexity on the held-out test sets to measure a model's ability to generalize to unseen data. Perplexity is a common criterion of clustering quality that does not require a priori categorizations and thus often used in the context of topic modelling. For a corpus of test documents $D^{test} = \{\boldsymbol{w}_d\}$, perplexity is the reciprocal geometric mean of the likelihood of this corpus given the model:

$$perplexity\left(D^{test}\right) = \exp\left\{-\frac{\sum_d \log p(\boldsymbol{w}_d)}{\sum_d N_d}\right\} .$$

Additionally, we evaluated all approaches in a classification setting for **WebKB**, **R8**, and **N20** . Specifically, we used the 7 first-level classes for **N20**, and all classes for **WebKB** and **R8**. Then, we used a multi-class linear support vector machine[5] to predict the class labels merely using the topic distributions of the documents as estimated by the learned LDA models. We report on the average accuracy achieved in a 5-fold cross-validation.

For the scaling experiment (**Q3**), we crawled our own **Wikipedia** (english) corpus of **D** = 2914700 randomly selected documents with **N** = 292941239 and a fixed vocabulary of **W** = 7686 words. We processed the crawled Wikipedia articles as done in [11] removing alls words from their vocabulary that did not appear in our articles. Additionally, we used a subset of **Grolier** encyclopedia articles, provided by Sam Roweis[6] with **D** = 30991 documents, **N** = 3484393 and a vocabulary of **W** = 15276 most common words, and the **NYTimes** news

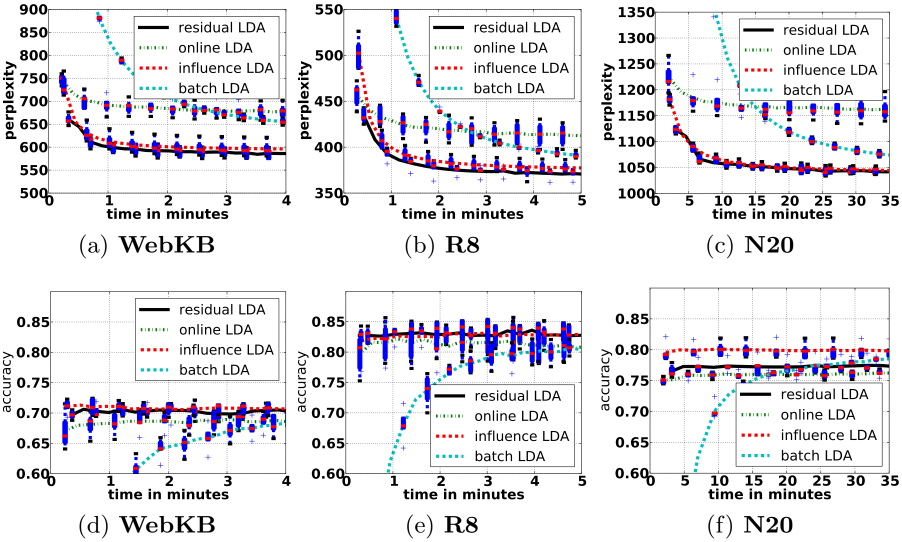
[2] http://www.cs.princeton.edu/~mdhoffma

[3] http://web.ist.utl.pt/~acardoso/datasets/

[4] http://www.kyb.tuebingen.mpg.de/bs/people/pgehler/rap/

[5] We used PyML http://pyml.sourceforge.net/ with the default settings.

[6] http://www.cs.nyu.edu/~roweis/data.html



(a) **WebKB** (b) **R8** (c) **N20**

(d) **WebKB** (e) **R8** (f) **N20**

Fig. 4. Small and medium scale experiments: Residual LDA outperforms batch and online LDA and is comparable to influence LDA. (a-c) Held-out perplexity (the lower, the better) as a function of CPU time (for K = 100). (d-e) Classification accuracy (the higher the better) as a function of CPU time taken to learn the low dimensional representation using different methods. (Best viewed in color)

articles provided by [6] with **D** = 300000 documents with **N** = 99542125 and a vocabulary of **W** = 102660 words. For **NYTimes**, after stoppword removal the vocabulary was reduced by keeping just words which appeared more than ten times, and additionally we excluded all multi token phrases. Again, we removed all articles with less than six words.

For all perplexity experiments we used 1000 documents as held-out test set for **Wikipedia** and **NYTimes**, and 500 documents for the remaining datasets. For all experiments, we set κ close to 0.5 as suggested in [11] and $\tau_0 = 4$ (determined by cross-validation on the training set of a subset of 25K webpages from **Wikipedia** and **N20** but used for all experiments). To set the batch size we used the following heuristic: $B = \frac{\mathbf{D}\cdot\|D\|_2}{\sum_i \|d_i\|_2}$, where $\| d_i \|_2$ resp. $\| D \|_2$ denotes the Frobenius norm of document i resp. the whole corpus. Intuitively, it measures how many documents are required to capture the important part of a corpus. We used fixed symmetric hyperparameters $\alpha = 0.01$ and $\eta = 0.01$ in all our experiments.

Q1, Q2: Small and Medium Datasets: The perplexity results for the small and medium datasets are summarized in Fig. 4 (a-c). In each experiment we computed perplexity after each time **D** documents were seen. Here, residual LDA finds sollutions in the range of influence scheduled LDA, and batch LDA's solution but with much less computation. Compared to online LDA on those

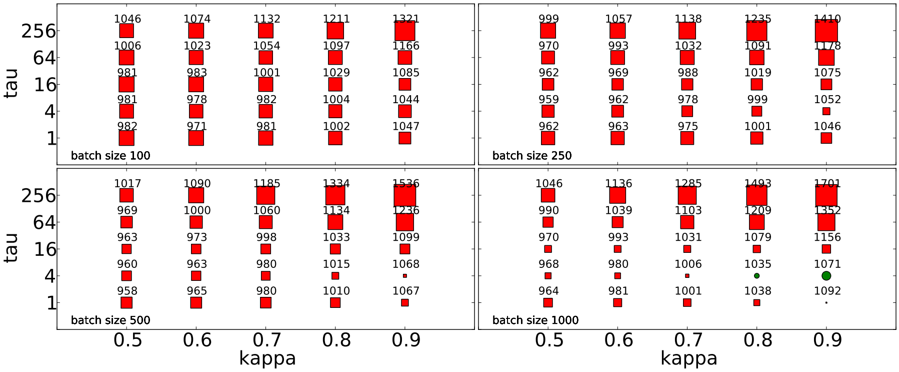
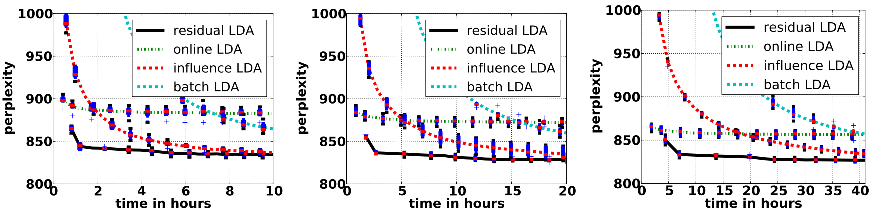


Fig. 5. Stability analysis: Hinton diagrams of held-out perplexity for different settings of the parameters $\tau_0 \in \{1, 4, 16, 64, 25\}$, $\kappa \in \{0.5, 0.6, \ldots, 0.9\}$ and $B \in \{100, 250, 500, 1000\}$ for a subset of 25K Wiki articles on a validation set of 1000 articles. Residual and online LDA were ran for 30 minutes. The achieved perplexity results were averaged over 5 reruns. The size of markers denote the difference in perplexity. Red squares indicate that residual LDA achieved lower perplexity, green circles that online LDA was the winner. The numbers are the absolute perplexities of the winners. Residual LDA is the winner in 98 of 100 parameter settings. (Best viewed in color)

datasets, residual LDA produces lower perplexity, i.e. it achieves significantly better performance. Here, we compared also residual LDA against online LDA on a subset of 25K Wikipedia articles for different parameter settings, as shown in Fig. 5 (we used 100 different parameter settings, for a total number of 500 runs). As one can see, the performance of residual LDA is more stable, and in 98% of cases better (red squares) than online LDA's perplexity (green circles).

Another point is, there is some question as to the meaningfulness of perplexity as a metric for comparing different topic models [3]. The accuracies of the classification experiments as summarized in Fig. 4 (d-e) provide a different metric. The results clearly show that residual LDA and isLDA yield predictive accuracies in the range of the batch model's with much less computation. They even outperform online LDA.

Q3: Scaling Experiments on Wikipedia: The results on the small datasets suggest that residual LDA and influence scheduled LDA are faster than batch LDA. To further investigate how they scale, we ran experiments on several Wikipedia corpora ranging from 250K to 1M documents. The results are summarized in Fig. 6. As one can see, isLDA indeed takes longer to converge than residual LDA. In contrast, residual LDA can handily analyze massive datasets and it can find topic models as good or better than those found with batch LDA or online LDA, and it converges much faster than isLDA. It is essentially always converged after the kink, which happens after about $2 \cdot \mathbf{D}$ documents were seen (again, here we measured the perplexity after each time $\mathbf{D}$ documents were seen). Also on the 3M Wikipedia dataset, see Fig. 1 (top), it produced better



(a) **Wiki 250K** (b) **Wiki 500K** (c) **Wiki 1M**

Fig. 6. Large scale experiments: Residual LDA outperforms all other methods. Held-out perplexity (the lower, the better) as a function of CPU time (for K = 100) for subsets of Wikipedia with different numbers **D** of articles. Results are averaged over five reruns. Residual LDA is essentially converged after the kink in its curve, which happens after about $2 \cdot \mathbf{D}$ documents were seen. (Best viewed in color)

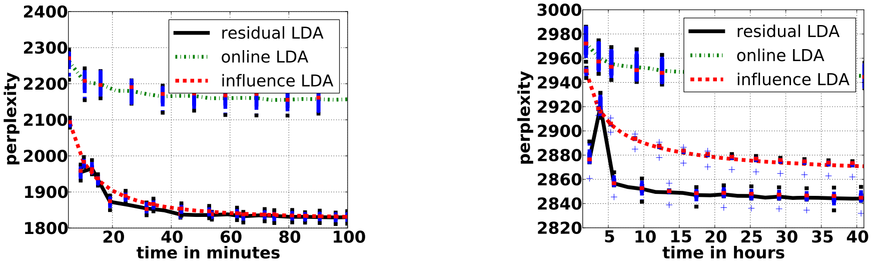
topics (in terms of perplexity) as those found by online LDA and batch LDA, and significantly faster. The topic evolution is shown in Fig. 1 (bottom) and illustrates that residual LDA's topic "university research" is qualitatively as good as the one found by batch LDA, but with a fraction of documents seen. Online LDA only starts to get similar to batch LDA. This is also supported by a smaller Hellinger distance to the batch LDA topic: residual LDA 0.12 vs. online LDA 0.53. Finally, to test the scaling behaviour with respect to both the vocabulary and dataset size, we ran experiments on **Grolier** and **NYTimes**. The results are summarized in Fig. 7. As one can see, residual LDA can handily analyse large corpora with large vocabulary size. For the smaller **Grolier**, residual LDA outperforms online LDA and is comparable to influence LDA. For the larger **NYTimes**, it outperforms online and influence LDA.

Thus, putting all experimental results together, we can clearly answer questions **Q1-Q3** affirmatively.

6 Conclusions

Triggered by the recent success stories of online LDA approach for the problem of inferring topics in growing document collections, we revisited batch LDA. We turned batch LDA into a quasi-online LDA approach that actively forms mini-batches of highly influential documents, called residual LDA. Specifically, residual LDA actively selects a subset of documents that exert a disproportionately large influence on the current residual to compute the next update. On several real-world datasets, including 3M articles from Wikipedia, we demonstrated that residual LDA can handily analyze massive document collections and find topic models as good or better than those found with batch VB and randomly scheduled VB, and significantly faster. In other words, *large residual* means *less work*.

Indeed, much remains to be done. One interesting avenue for future work is to employ influence schedules in parallel LDA approaches. Another is to discard



(a) **Grolier** (b) **NYTimes**

Fig. 7. Residual LDA can handily analyse large corpora with large vocabulary size. Held-out perplexity (the lower, the better) as a function of CPU time (for K = 100) for **Grolier** and **NYTimes** datasets. Results are averaged over five reruns. For the smaller dataset residual LDA outperforms online LDA and is comparable to influence LDA. For the larger dataset, residual LDA outperforms both. (Best viewed in color)

documents during learning (based on inference). Most interesting, however, is to develop strong theoretical guarantees for residual LDA and to transfer them to the general stochastic optimization case. That this is possible shows a connection to the recent work of Yi *et al.* [24]. We assume a fitness function and want to maximize the expected fitness under the search distribution. Following Yi *et al.*, we assume that gradient entries are Gaussian distributed and compute the covariance over all gradients. Assuming a uniform fitness over gradients and computing the gradient with the steepest ascent of the expected fitness, Yi *et al.*'s results show that the gradient of the likelihood of the search direction is the most influential one. This measure, however, depends on the norm of the corresponding gradient, which is exactly what residual LDA employs.

Acknowledgements. The authors thank the anonymous reviewers for their comments and their colleagues Christian Bauckhage and Anja Pilz for valuable discussions. They also thank Anja Pilz for crawling the Wikipedia dataset and Matt Hoffman for making the online LDA Python code publically available. MW and KK were supported by the Fraunhofer ATTRACT fellowship STREAM.

References

1. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
2. Canini, K., Shi, L., Griffiths, T.: Online inference of topics with latent dirichlet allocation. In: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, AISTATS 2009 (2009)
3. Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., Blei, D.: Reading tea leaves: How humans interpret topic models. In: Proceeding of NIPS (2009)
4. Ding, C., Li, T., Peng, W.: NMF and PLSI: Equivalence and a Hybrid Algorithm. In: Proc. SIGIR (2006)

5. Drineas, P., Kannan, R., Mahoney, M.: Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. SIAM Journal of Computing 36, 184–206 (2006)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml
7. Frieze, A., Kannan, R., Vempala, S.: Fast monte-carlo algorithms for finding lowrank approximations. Journal of the ACM 51(6), 1025–1041 (2004)
8. Gaussier, E., Goutte, C.: Relations between PLSA and NMF and Implications. In: Proc. SIGIR (2005)
9. Gehler, P., Holub, A., Welling, M.: The rate adapting poisson model for information retrieval and object recognition. In: Proceedings of ICML, pp. 337–344 (2006)
10. Girolami, M., Kaban, A.: On an Equivalence between PLSI and LDA. In: Proc. SIGIR (2003)
11. Hoffman, M., Blei, D., Bach, F.: Online learning for latent dirichlet allocation. In: Proceedings of Neural Information Processing Systems (NIPS 2010) (2010)
12. Hofmann, T.: Probabilistic latent semantic indexing. Research and Development in Information Retrieval, pp. 50–57 (1999)
13. Hofmann, T., Buhmann, J.: Active data clustering. In: Proceedings of NIPS (1997)
14. Mahoney, M., Drineas, P.: Cur matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 106(3), 697–703 (2009)
15. Newman, D., Asuncion, A., Smyth, P., Welling, M.: Distributed algorithms for topic models. Journal of Machine Learning Research 10, 1801–1828 (2009)
16. Sato, I., Kurihara, K., Nakagawa, H.: Deterministic single-pass algorithm for lda. In: Proceedings of Neural Information Processing Systems, NIPS 2010 (2010)
17. Settles, B.: Active learning literature survey. Tech. Rep. 1648, University of Wisconsin-Madison (2010)
18. Smola, A., Narayanamurthy, S.: An architecture for parallel topic models. PVLDB 3(1), 703–710 (2010)
19. Sun, J., Xie, Y., Zhang, H., Faloutsos, C.: Less is more: Sparse graph mining with compact matrix decomposition. Statistical Analysis and Data Mining 1(1), 6–22 (2008)
20. Wallach, H., Mimno, D., McCallum, A.: Rethinking lda: Why priors matter. In: Advances in Neural Information Processing Systems, vol. 22, pp. 1973–1981 (2009)
21. Wang, X., Davidson, I.: Active spectral clustering. In: Proceedings of the IEEE International Conference on Data Mining (ICDM 2010) (2010)
22. Yan, F., Xu, N., Qi, Y.: Parallel inference for latent dirichlet allocation on graphics processing units. In: Proceedings of NIPS (2009)
23. Yao, L., Mimno, D., McCallum, A.: Efficient methods for topic model inference on streaming document collections. In: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 937–946 (2009)
24. Yi, S., Wierstra, D., Schaul, T., Schmidhuber, J.: Stochastic search using the natural gradient. In: Proceedings of ICML, p. 146 (2009)

A Community-Based Pseudolikelihood Approach for Relationship Labeling in Social Networks

Huaiyu Wan*, Youfang Lin*, Zhihao Wu, and Houkuan Huang

School of Computer and Information Technology,
Beijing Jiaotong University, Beijing 100044, China
{huaiyuwan,yflin,zhihaowu,hkhuang}@bjtu.edu.cn

Abstract. A social network consists of people (or other social entities) connected by a set of social relationships. Awareness of the relationship types is very helpful for us to understand the structure and the characteristics of the social network. Traditional classifiers are not accurate enough for relationship labeling since they assume that all the labels are independent and identically distributed. A relational probabilistic model, relational Markov networks (RMNs), is introduced to labeling relationships, but the inefficient parameter estimation makes it difficult to deploy in large-scale social networks. In this paper, we propose a community-based pseudolikelihood (CBPL) approach for relationship labeling. The community structure of a social network is used to assist in constructing the conditional random field, and this makes our approach reasonable and accurate. In addition, the computational simplicity of pseudolikelihood effectively resolves the time complexity problem which RMNs are suffering. We apply our approach on two real-world social networks, one is a terrorist relation network and the other is a phone call network we collected from encrypted call detail records. In our experiments, for avoiding losing links while splitting a closely connected social network into separate training and test subsets, we split the datasets according to the links rather than the individuals. The experimental results show that our approach performs well in terms of accuracy and efficiency.

Keywords: Social networks, Relationship labeling, Community structure, Pseudolikelihood, Conditional random fields.

1 Introduction

Social networks are a ubiquitous paradigm of human interactions in real world. People in social networks are connected to each other by different types of relationships, such as family, friendship, co-working, collaboration, contact, etc. Given a snapshot of a social network with content and link structure, can we infer the types of the relationships between the individuals? This question can be formalized as the *relationship labeling problem*. Labeling relationships is one of the most significant problems in the research of social networks. For instance,

* Corresponding authors.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 491–505, 2011.

in a criminal network, the labels of the relationships between the criminals can help the police to discover regular patterns about the organization and operation of the criminal group. Considering another example of a social network which consists of all the mobile users in a particular region, knowing the type of the relationship between each pair of communicated users can greatly help the mobile service providers to develop targeted marketing strategies.

In many real world applications, a common situation of the relationship labeling task is that, some small part of the relationships in a social network can be directly labeled in some way, while the others cannot be labeled directly and inference is needed. This is so-called *within-network learning*. For example, in a criminal network, some relationships between the criminals can be labeled through the police investigation, while the others must be labeled in other ways. Similarly, in a mobile phone call network, a few relationships between the communicated users can be labeled through the service packages (i.e., family packages or group packages) ordered by the users, but more other relationships cannot be labeled in this way.

The basic idea for classifying the relationships is employing traditional classifiers in the flat setting by using the content attributes of the relationships, where all the labels are assumed to be independent and identically distributed (IID). However, this completely ignores the rich information of the link structure, which generally reflects the common patterns of interactions among the individuals in a social network. Therefore, Taskar et al. [1] and Zhao et al. [2] adopt relational Markov networks (RMNs) [3], a statistical relational learning framework, to classify the relationship labels in webpage networks and terrorist networks respectively, but the inefficient parameter estimation makes it very difficult to deploy this model in large-scale social networks. In addition, the definition of the *relational clique templates* will greatly affect the prediction accuracy of RMNs in practice.

In this paper, we propose a community-based pseudolikelihood (CBPL) approach to labeling relationships in social networks. In our approach, we use the community structure of a social network to assist in constructing the conditional random field (CRF). As we know, community structure is one of the most important properties of complex networks [4]. According to the notion of “birds of a feather flock together”, individuals in the same community tend to have the same type, and thus relationships starting from the same individual and terminating in the same community tend to have the same label. Fig. 1 depicts an example fragment of a terrorist social network [2] from the Profiles in Terror (PIT) knowledge base. A dashed ellipse indicates a community, and the various relationship types are distinguished by different line styles and colors. This figure clearly illustrates the correlation between the relationship labels and the community structure of the network.

As an efficient alternative of likelihood, the pseudolikelihood measure [5] is often employed to approximate the joint probability distribution of a collection of random variables with a set of conditional probability distributions (CPDs). This technique effectively resolves the time complexity problem which the RMN

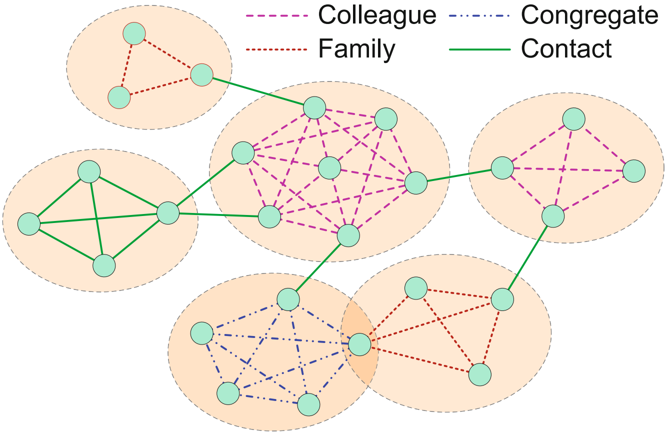


Fig. 1. An example fragment of a terrorist social network

model is suffering and makes our approach more efficient for handling large-scale social networks.

We present experiments using our approach on two real-world social networks, one is a terrorist social network [2] and the other is a phone call network we collected from encrypted call detail records (CDRs). A problem we often encounter in the experiments on within-networks is that splitting a closely connected network into separate training and test subsets will lose the information of the links that go from one subset to another. In our experiments, for avoiding losing such information, we split the datasets according to the links (i.e., relationships) rather than the individuals. This ensures that each link will appear in either a training subset or a test subset. The experimental results show that our proposed approach is much more accurate and efficient than the RMN approach on the task of relationship labeling in social networks.

The rest of the paper is organized as follows. The next section provides a brief discussion of related work. Section 3 presents our approach in detail, followed by the experimental evaluations in section 4. Finally, we give the conclusion and future work in section 5.

2 Related Work

As discussed earlier, it is not accurate enough for relationship labeling in social networks by only using the content attributes, since the rich information of link structures is completely ignored. Taskar et al. [1] treat the relationship labeling problem as a task of *link prediction* and use RMNs to predict the labels of the links between the Computer Science department webpages from three universities in American. RMNs [3,6] are a joint probabilistic modeling framework for an entire collection of related entities building on undirected graph models, and provide a form of collective classification in which we can simultaneously decide on the class labels of all the relationships together rather than classify each relationship separately. Zhao et al. [2] pay their attention to the counter-terrorism domain. They extract a terrorist social network from the PIT knowledge base (http://profilesinterror.mindswap.org/) and try to predict the types of the relationships between the terrorists. RMNs are also employed for their experiments. Since two terrorists can be related in multiple relationships, multi-label classification is considered.

There are two problems with using RMNs for labeling relationships in social networks:

- The computational complexity of learning RMNs is very high. That is because multiple rounds of approximate inference (e.g., loopy belief propagation) are required over the entire dataset. Especially in our relationship labeling task, the number of relationships is the squared magnitude of the number of individuals in a social network. So the training time is usually unacceptable if the scale of the social network is too large. In addition, numerous short, closed loops in large-scale RMNs usually cause the belief propagation algorithm to return a poor approximation and even not to converge to a stationary state.
- The prediction accuracy of the RMN model directly depends on the definition of relational clique templates. For relationship labeling, the most direct method is to construct dyad cliques for any pair of relationships which have a common individual and triad cliques for any triple of relationships which connected end to end. However, this will not always be correct in case the individuals in the same clique are not of the same type. So the labeling accuracy will be affected to some extent.

In this work, we propose to use the pseudolikelihood technique to estimate the labels of relationships in social networks. Since pseudolikelihood can only capture the local dependencies and ignores the indirect effects between the non-neighboring variables, it may lose some accuracy in practice. However, we must consider a tradeoff between the prediction accuracy and the computational complexity in relational learning, especially in case the scale of a social network is very large. Actually, as an efficient alternative measure of likelihood, pseudolikelihood has been successfully used in the relational learning field. Richardson and Domingos [7] proposed optimizing a pseudolikelihood measure to learning an Markov logic network (MLN) [8], where the full joint distribution is approximated as a product of each variable's probability conditioned on its Markov blanket. Relational dependency networks (RDNs) [9], an undirected graphical model for relational data introduced by Neville and Jensen, approximate the full joint distribution of an entire dataset with a set of CPDs based pseudolikelihood techniques. Xiang and Neville [10] developed a semi-supervised pseudolikelihood expectation maximization (PL-EM) algorithm, which has been demonstrated to be effective in within-network learning.

Community structure is used in our proposed approach to assist in constructing the CRF of a social network, and we believe that this will amend the limitation of the pseudolikelihood measure and make our approach more reasonable and accurate. The property of community structure has been successfully used to describe the dependencies between the variables in relational data. Neville and Jensen [11] proposed latent group model (LGM), which posits that the class labels of the objects in a relational dataset are related to their group (or community) types. Within each group, the class labels are conditionally independent given the group type. Another relational model similar to LGM is the latent social dimension (LSD) model [12], which extracts latent social dimensions of objects from a modularity matrix defined on the modularity measure [13] and then considers these dimensions as normal features of objects for prediction tasks. The above two models demonstrate that the community structure is really very helpful for relational learning.

Wang et al. [14] studied the mining of advisor-advisee relationship from research publication networks and proposed a time-constrained probabilistic factor graph (TPFG) model, which is an unsupervised approach and suitable for the situation that not any labeled relationships can be used as supervised information.

3 Approach

We use a graph $G = (V, E)$ to represent a social network, where V is the set of individuals, and E is the set of links (i.e., relationships) between the individuals. Suppose that the content attributes of the individuals and the link structure are known, and some relationship labels are observed, then our task is to predict the remaining unobserved labels.

In relationship labeling, we need to make relationships the first-class citizens. Given an instantiation $\mathcal{I}$ of our schema, the pseudolikelihood $PL(\mathcal{I})$ is the product of the conditional probability of each variable $Y_i (i \leq |E|)$ given its Markov blanket $MB(Y_i)$. So we need to specify the neighboring relationships for each relationship $e \in E$, i.e., to construct a Conditional random field (CRF) for all the relationships over the entire social network. CRFs are undirected graphical models that were developed for labeling sequence data [15], and are well suited for discriminative training, which generally provides significant improvements in classification accuracy over generative training given sufficient training examples [16]. As discussed in section 2, simply letting two relationships which have a common individual be the neighboring nodes in the CRF will not always be appropriate. Consequently, we resort to using the community structure of the social network to assist in constructing the CRF. For maintaining the structural integrity of the social network, we detect its communities over the entire dataset, rather than on the training and test subsets respectively. After the construction of the CRF, the pseudolikelihood model is trained and tested on the training and test subsets respectively. The detailed steps of our approach are as follows.

3.1 Step 1: Community Detection

We first run a community detection algorithm on the graph G of the social network instantiation $\mathcal{I}$. According to whether intersecting communities are generated, community detection algorithms can be divided into non-overlapping and overlapping algorithms. *Non-overlapping community detection* supposes that every individual only belongs to a single community, while *overlapping community detection* considers the natural phenomenon that one person may belong to multiple groups in real world, thus allows an individual to belong to more than one communities. Many sophisticated community detection algorithms have been developed in recent years and Fortunato [17] provides a detailed summary.

People can select non-overlapping or overlapping community detection algorithms according to the overlapping property of a social network, i.e., if an individual can belong to multiple communities. In this paper, for comparing the performance of different community detection algorithms, we use both non-overlapping and overlapping algorithms. It is noted that, a community detection algorithm is needed to be executed only once for a social network instantiation, no matter how to split the training and test subsets subsequently.

3.2 Step 2: Conditional Random Field Construction

We construct the CRF based on both the original social network and the community results obtained in Step 1. The principle is very simple: we treat the relationships in the original social network as the nodes in the CRF, and then establish a link between any pair of relationships if they start from the same individual and terminate in the same community. Fig. 2 lists all the possible community structures of any two neighboring relationships, in which (c) and (f) are overlapping communities. Concretely, we establish a link between any pair of relationships with a community structure like (a), (b), or (c) in the figure, while the situations like (d), (e), and (f) are ignored. Finally, we add the content attributes into the CRF as the evidence for each relationship.

We use $\mathcal{F} = (\mathbf{Y}, \mathbf{x}, \mathbf{r})$ to denote the CRF of the instantiation $\mathcal{I}$, where $\mathbf{Y}$ is the set of label variables and $\mathbf{x}$ is the set of content attributes and $\mathbf{r}$ is the set of links between the relationships. After the construction of the CRF, the Markov blanket of each label variable is determined.

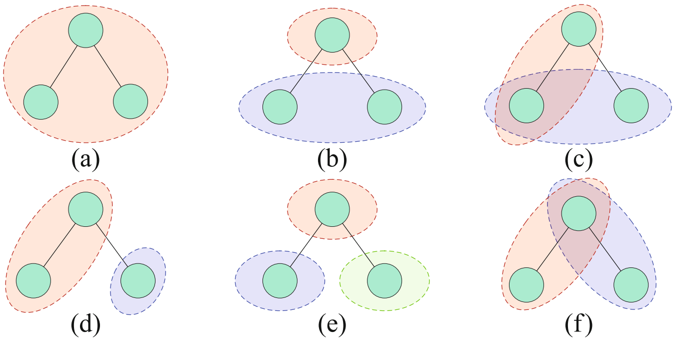


Fig. 2. All the possible community structures of two neighboring relationships

3.3 Step 3: The Pseudolikelihood Model

Given the CRF $\mathcal{F} = (\mathbf{Y}, \mathbf{x}, \mathbf{r})$, for each label Y_i, pseudolikelihood models use a local CPD $P(y_i|MB(y_i))$ to represent the conditional probability of the label value y_i, where $MB(y_i)$ is the state of the Markov blanket of Y_i in the data. It is noted that, for simplifying the representation, we let the Markov blanket $MB(Y_i)$ contain not only the neighboring label variables but also the content attributes of Y_i. We maximize the following pseudolikelihood

$$P(\mathbf{y}|\mathbf{x}, \mathbf{r}) = \prod_{i=1}^{n} P(y_i|MB(y_i)), \tag{1}$$

where n is the number of label variables in $\mathcal{F}$.

Let each pair of neighboring nodes in $\mathcal{F}$ to be a *clique* with a potential ϕ, according to the fundamental theorem of Markov random fields [18], the conditional probability $P(y_i|MB(y_i))$ can be factorized over all of the cliques:

$$P(y_i|MB(y_i)) = \frac{1}{Z_i(\mathbf{x}_i, \mathbf{r}_i)} \prod_{v_j \in MB(y_i)} \phi(y_i, v_j), \tag{2}$$

where Z_i is the local partition function (or normalization constant) given by

$$Z_i(\mathbf{x}_i, \mathbf{r}_i) = \sum_{y'_i} \prod_{v_j \in MB(y'_i)} \phi(y'_i, v_j). \tag{3}$$

Therefore, computing pseudolikelihood is very efficient because the local partition function is simply a sum over a single variable.

The potential is often represented by a log linear combination of a set of features:

$$\begin{aligned} \phi(y_i, v_j) &= \exp\{\sum_k w_k f_k(y_i, v_j)\} \\ &= \exp\{\mathbf{w} \cdot \mathbf{f}(y_i, v_j)\}, \end{aligned} \tag{4}$$

where w_k is the weight of the feature f_k.

Parameter Learning. The goal of parameter learning is to determine the weights of the features in the pseudolikelihood model. Maximum a posterior (MAP) training is used to learn the pseudolikelihood model. To avoid overfitting, we assume the prior of the weights $\mathbf{w}$ is a zero-mean Gaussian. The log of the MAP objective function is as follows:

$$\begin{aligned} PL(\mathcal{I}, \mathbf{w}) &= \log \Big(P(\mathbf{y}|\mathbf{x}, \mathbf{r}) \prod_k P(w_k) \Big) \\ &= \log P(\mathbf{y}|\mathbf{x}, \mathbf{r}) + \sum_k \frac{-w_k^2}{2\sigma^2} - \log \sqrt{2\pi\sigma^2} \\ &= \sum_{i=1}^{n} \Big(\sum_{v_j \in MB(y_i)} \mathbf{w} \cdot \mathbf{f}(y_i, v_j) - \log Z_i \Big) - \frac{||\mathbf{w}||_2^2}{2\sigma^2} + C. \end{aligned} \tag{5}$$

$PL(\mathcal{I}, \mathbf{w})$ is a concave function and we can estimate the parameters $\mathbf{w}$ by maximizing the log-pseudolikelihood by using a variety of gradient-based optimization algorithms, such as conjugate gradient or quasi-Newton. For computing the gradient, the derivative of $PL(\mathcal{I}, \mathbf{w})$ with respect to w_m is given by

$$\frac{\partial PL(\mathcal{I}, \mathbf{w})}{\partial w_k} = \sum_{i=1}^{n} \sum_{v_j \in MB(y_i)} \Big\{ f_k(y_i, v_j) - E_{P_\mathbf{w}} \left[f_k(y_i, v_j) \right] \Big\} - \frac{w_k}{\sigma^2}, \qquad (6)$$

where the expected feature values is related to $P_\mathbf{w}$:

$$E_{P_\mathbf{w}} \left[f_k(y_i, v_j) \right] = \sum_{y_i'} \Big\{ f_k(y_i', v_j) P_\mathbf{w}(y_i', v_j) \Big\}. \qquad (7)$$

The time complexity of computing the gradient in equation (6) is $O(n*r)$, where n is the number of label variables and r is the number of links in the CRF $\mathcal{F}$. Comparatively, the complexity of learning an RMN model is much higher because approximate inference is required, and generally the complexities of approximate inference algorithms are very high. For example, the complexity of loopy belief propagation is $O(m * n * r^2)$, where m is the number of iterations. Furthermore, the use of community structure also reduces some computational cost, since the removal of some unreasonable links among the relationships makes the CRF a little sparser. In conclusion, our CBPL model is much more efficient then the RMN model in terms of computational complexity.

Inference. Given the observed content attributes $\mathbf{x}$ and the parameters $\mathbf{w}$, the task of inference is to determine the relationship labels. As we know, loopy belief propagation (LBP) [19,20] is often used to inference CRFs. Our inference algorithm is very similar to LBP, which estimates the marginal distribution of each label variable approximately by sending local messages through the graph structure of the model. We initialize the marginals and values of the label variables by using only the content attributes, and then update them iteratively with the state of their Markov blankets at the previous time, until each of the variables does not change any more. The detailed procedures of the inference algorithm are presented in Algorithm 1.

4 Experiments

In this section, we present a set of experiments to evaluate our CBPL approach. We performed relationship labeling on two real-world datasets, and compared the performance of our approach with that of the RMN model. The results of the content-only relationship labeling were taken as our baseline.

4.1 Datasets

TerroristRel[1]**.** It is a public dataset contributed by Zhao et al. [2] and collected from the PIT knowledge base. The dataset contains information about terrorists

[1] http://www.cs.umd.edu/projects/linqs/projects/lbc/

Algorithm 1. CBPL-Inference

```
Input: content attributes $\mathbf{x}$, links $\mathbf{r}$, parameters $\mathbf{w}$
Output: labels $\mathbf{Y}$
// initiation:
foreach label variable $Y_i$ do
    foreach value $y_i$ do
        // initialize the local CPD by using only the content attributes
        $P^{(0)}(y_i|MB(y_i)) \leftarrow \prod_{x_j \in MB(y_i)} \phi(y_i, x_j)/Z_i(\mathbf{x}_i, \mathbf{r}_i)$;
    end
    $Y_i^{(0)} \leftarrow \arg\max_{y_i} P^{(0)}(y_i|MB(y_i))$;
end
// iteration:
repeat
    foreach label variable $Y_i$ do
        foreach value $y_i$ do
            // update the local CPD by using the state of $MB(y_i)$ at $t$-1
            $P^{(t)}(y_i|MB(y_i)) \leftarrow \prod_{v_j^{(t-1)} \in MB(y_i)} \phi(y_i, v_j^{(t-1)})/Z_i(\mathbf{x}_i, \mathbf{r}_i)$;
        end
        $Y_i^{(t)} \leftarrow \arg\max_{y_i} P^{(t)}(y_i|MB(y_i))$;
    end
until each variable $Y_i$ satisfies $Y_i^{(t)} = Y_i^{(t-1)}$ ;
```

and their relationships and was designed for labeling the relationships between the terrorists. It consists of 244 terrorists and 840 relationships between them. Each relationship is described by a 0/1-valued vector where each component indicates the absence/presence of one of the total of 1224 distinct features. Each relationship can be assigned one or more labels within the labels of Family (16.0%), Colleague (54.2%), Congregate (12.4%), and Contact (17.4%).

PhoneCallNet. We collected a phone call network from the encrypted CDRs of the mobile users in an area in northern China obtained from one of the largest mobile service providers in China. The CDRs recorded all the phone call and short message (SM) events occurred within about 15 days in July 2010. We extracted 1623 mobile users and 4295 distinct relationships between them. For each relationship, we derived 9 statistical properties (as listed in table 1) from the CDRs and took them as the content attributes. All these statistical properties were normalized by being divided by the total call count, the total call duration, and the total SM count, respectively.

Manually labeling the relationships in this dataset for testing was not an easy task. We used the service packages provided by the mobile service provider to label the relationships. Actually, all the instances in our dataset were collected among the users who ordered at least one family or group package. Then the relationships between the users who ordered the same family package were labeled with Family (22.0%), and the relationships between the users who ordered the same group package were labeled with Group (63.3%), and the remaining relationships were labeled with Common (14.7%).

Table 1. The statistical properties of the relationships in the PhoneCallNet dataset

Feature	**Description**
call_busy_count	the call count between 08:30 and 17:30 h on weekdays
call_free_count	the call count between 17:30 and 08:30 h on weekdays
call_weekend_count	the call count on weekend
call_busy_duration	the call duration between 08:30 and 17:30 h on weekdays
call_free_duration	the call duration between 17:30 and 08:30 h on weekdays
call_weekend_duration	the call duration on weekend
sm_busy_count	the SM count between 08:30 and 17:30 h on weekdays
sm_free_count	the SM count between 17:30 and 08:30 h on weekdays
sm_weekend_count	the SM count on weekend

4.2 Experimental Setup

Baseline. Our approach is a link-based classification method in the relational learning field, so we focused on the comparison of our method with a representational relational learning model (i.e., RMNs). However, for achieving more information, we take the results of the content-only relationship labeling as our baseline. Traditional classifiers, such as naïve Bayes, logistic regression or SVM, can be used to perform content-only relationship labeling. In our experiments, we chose to use the conditional Markov Networks [3] in the flat setting as a representative.

Community Detection Algorithms. In our CBPL approach, we respectively employed the Infomap algorithm[2] [21] as the non-overlapping community detection algorithm and the Greedy Clique Expansion (GCE) algorithm[3] [22] as the overlapping one. The Infomap algorithm, proposed by Rosvall and Bergstrom, finds the best cluster structure of a graph by optimally compressing the information describing the probability flow of random walk and has a complexity that is essentially linear in the size of the graph (i.e., $O(|E|)$). It is considered as one of the best community detection algorithms so far [23]. The GCE algorithm is one of the newest overlapping community detection algorithms to detect the intersecting communities in social networks. It identifies distinct cliques as seeds and expands these seeds by greedily optimizing a local community fitness function, and then finally accepts only those communities that are not near-duplicates of communities that have already been accepted.

Relational Clique Templates. For the RMN model, we defined two relational clique templates as follows: 1) dyad cliques for any pair of relationships which have a common individual; and 2) triad cliques for any triple of relationships which connected end to end.

[2] http://www.tp.umu.se/~rosvall/code.html

[3] http://sites.google.com/site/greedycliqueexpansion/

Feature Functions. Feature functions are needed to be defined for both our CBPL approach and the RMN model. All the cliques can be divided into two categories: the cliques which consist of one label variable and one content attribute belong to the category of *evidence cliques*, while the cliques which contain only label variables belong to the category of *compatibility cliques*. For the evidence cliques, we defined a feature with the form of $f(y_i, x_j) = y_i x_j$, where $y_i = \pm 1$, and $x_j \in \{1, 0\}$ for the TerroristRel dataset and $x_j \in [0, 1]$ for the PhoneCallNet dataset. For a compatibility clique, we simply use a single feature to track whether all of its labels are the same.

Multi-label Classification. For the TerroristRel dataset, since two terrorists can be related in multiple relationships, multi-label classification was considered. We used a simple way that learned and tested a binary (one-against-rest) classifier for each of the four labels respectively, and then computed their average performance.

Dataset Splitting. A problem we often encounter in within-network learning is that directly splitting a closely connected network into separate training and test subsets would lose a lot of links which go from one subset to another. Consequently, we split our datasets according to the relationships rather than the individuals. Specifically, we put each relationship into the training or test subset with a certain probability, and in this case an individual might be in both the two subsets simultaneously. For evaluating the performance of our approach in different proportions of the observed labels, we randomly chose 10%, 20%, 30%, 40%, and 50% relationships, respectively, as the observed data for training, and the remaining relationships were used for testing.

4.3 Results and Discussions

The relationship labeling accuracies of the various approaches for the two datasets are shown in Fig. 3 and Fig. 4 respectively. Each experiment in this study was repeated 10 times and the results were averaged. From the two figures we can see:

1) The prediction accuracies of the relational approaches (whether the RMN model or our CBPL approach) are much better than that of the content-only approach. This demonstrates that the link structure is a very important source of information, and the relational approaches are able to learning social networks effectively by integrating information from content attributes of individuals as well as the links between them.
2) Our CBPL approach outperforms the RMN model (increases the labeling accuracies by around 2-4% for the TerroristRel dataset and 3-5% for the PhoneCallNet dataset respectively). And this demonstrates that the community structure is really very helpful for relationship labeling in social networks. Although the pseudolikelihood technique ignores the indirect dependencies and may lose some accuracy, the use of community structures makes up for this deficiency to a large extent by describing the local direct dependencies more reasonable and more accurate.

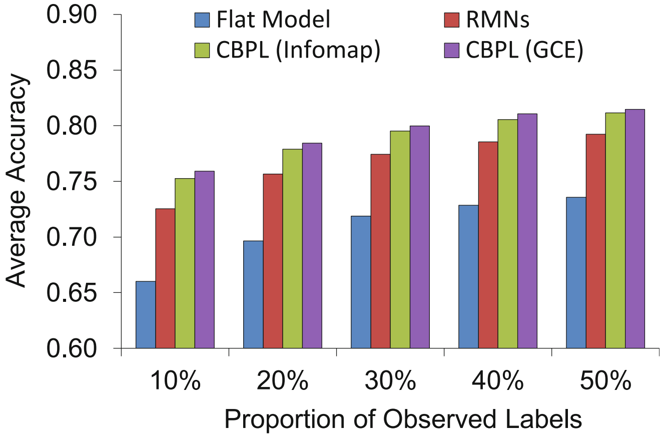


Fig. 3. Average classification accuracies for the TerroristRel dataset

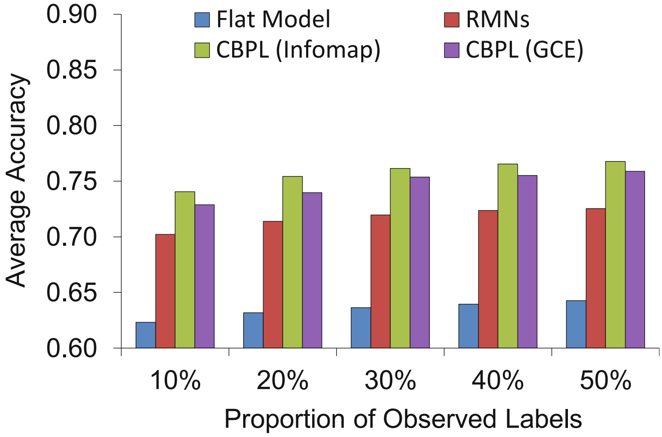


Fig. 4. Average prediction accuracies for the PhoneCallNet dataset

3) For the TerroristRel dataset, the CBPL approach based on overlapping community detection slightly outperforms the one based on non-overlapping community detection. The situation is just the opposite for the PhoneCallNet dataset. We believe this is due to the different community structure properties of the two datasets. That is, the overlapping nature of the communities in the TerroristRel dataset is quite strong so the CBPL approach based on the GCE algorithm performs better, and on the contrary, the overlapping nature of the PhoneCallNet dataset is weak so the CBPL approach based on the Infomap algorithm performs better.

4) Because the number of content attributes of the PhoneCallNet dataset is fewer (just 9 statistical properties), the increases of the labeling accuracies

along with the proportion of observed labels are not obvious for the various approaches. Therefore, if more features about the relationships between the communicated users were observed, the prediction accuracies could be higher.

The average training times of the various approaches along the proportion of observed labels for the two datasets are shown in Table 2 and Table 3. From the tables we see that: 1) our CBPL approach, whose training speeds are almost of the same order of magnitude as those of the flat model, is much more efficient than RMNs; and 2) the increasing rates of the training times of RMNs become much higher along with the growth of the proportion of observed labels, while those of CBPL are almost nearly linear.

Table 2. Average Training Times (Seconds) for the TerroristRel Dataset. All the results were computed on a PC with CPU 3.0 GHz and 2 GB RAM. Note that the time of community detection was not contained in the training times of our CBPL approach, since it is only in several milliseconds and very short comparing with the time of learning the pseudolikelihood model.

Approach	Proportion of Observed Labels				
	10%	20%	30%	40%	50%
Flat Model	0.81	2.06	3.64	7.79	11.83
RMNs	4.49	25.86	96.41	289.05	820.60
CBPL (Infomap)	2.01	6.02	15.48	34.85	51.27
CBPL (GCE)	2.53	7.91	18.96	39.58	58.73

Table 3. Average Training Times (Seconds) for the PhoneCallNet Dataset

Approach	Proportion of Observed Labels				
	10%	20%	30%	40%	50%
Flat Model	0.79	1.64	2.45	3.31	5.87
RMNs	6.53	33.62	133.84	437.76	1362.54
CBPL (Infomap)	1.26	4.92	12.85	27.41	46.05
CBPL (GCE)	1.67	5.63	15.39	32.27	53.72

5 Conclusion and Future Work

In this paper we studied the problem of relationship labeling in social networks and proposed a community-based pseudolikelihood approach. In our approach we use the community structure, one of the most important properties of complex networks, to assist us in constructing the conditional random field and the pseudolikelihood measure is employed to approximate the joint probability distribution of a collection of relationship label variables. The use of community structures makes our approach more reasonable and more accurate to describe the dependencies between the variables in relational data, while the pseudolikelihood technique effectively resolves time complexity problem which the RMN model is suffering and makes our approach much easier to deploy in large-scale social networks.

We applied our CBPL approach to the task of relationship labeling on some real-world social networks, including a terrorist relation network and a mobile user phone call network. The experimental results showed that our approach performs well in terms of accuracy and efficiency.

There are still some works can be improved in the future. Firstly, this paper first proposes using community structure to improve link-based classification and the experiments show that the community information is really useful in practice, but the quantification of the community information should be researched further. Secondly, since our proposed approach is a supervised learning technique, fully-labeled data is needed for training the model and we must split entire social networks into separate training and test subsets in practice. Actually, semi-supervised learning may be more suitable for such partially labeled within-networks. Therefore, the development of semi-supervised community-based relationship labeling methods will be one of our future research topics. Lastly, this paper is focused on the relationship labeling problem of ordinary social networks which consist of individuals as well as the relationships between them. The generalization of our approach to multipartite or even multimode networks could also be one of our future works.

Acknowledgments. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported in part by National Nature Science Foundation of China (Grant No. 60905029), Beijing Natural Science Foundation (Grant No. 4112046) and the Fundamental Research Funds for the Central Universities of China.

References

1. Taskar, B., Wong, M.F., Abbeel, P., Koller, D.: Link prediction in relational data. In: Neural Information Processing Systems 2003, pp. 659–666. The MIT Press, Cambridge (2004)
2. Zhao, B., Sen, P., Getoor, L.: Entity and relationship labeling in affiliation networks. In: ICML 2006 Workshop on Statistical Network Analysis: Models, Issues, and New Directions (2006)
3. Taskar, B., Abbeel, B., Koller, D.: Discriminative probabilistic models for relational data. In: 18th Conference on Uncertainty in Artificial Intelligence, pp. 485–492. Morgan Kaufmann, San Francisco (2002)
4. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS 99(12), 7821–7826 (2002)
5. Besag, J.: Statistical analysis of non-lattice data. The Statistician 24(3), 179–195 (1975)
6. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007)

7. Richardson, M., Domingos, P.: Markov logic networks. Technical report, Department of Computer Science and Engineering, University of Washington (2004)
8. Domingos, P., Richardson, M.: Markov logic: a unifying framework for statistical relational learning. In: ICML 2004 Workshop on Statistical Relational Learning and its Connections to Other Fields, pp. 49–54. IMLS, Washington, DC (2004)
9. Neville, J., Jensen, D.: Collective classification with relational dependency networks. In: KDD 2003 Workshop on Multi-Relational Data Mining, pp. 77–91 (2003)
10. Xiang, R., Neville, J.: Pseudolikelihood EM for within-network relational learning. In: 8th IEEE International Conference on Data Mining, pp. 1103–1108. IEEE Computer Society, Washington, DC (2008)
11. Neville, J., Jensen, D.: Leveraging relational autocorrelation with latent group models. In: 5th IEEE International Conference on Data Mining, pp. 322–329. IEEE Computer Society, Washington, DC (2005)
12. Tang, L., Liu, H.: Relational learning via latent social dimensions. In: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 817–826. ACM Press, New York (2009)
13. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E 69(2), 026113 (2004)
14. Wang, C., Han, J., Jia, Y., Tang, J., Zhang, D., Yu, Y., Guo, J.: Mining advisor-advisee relationships from research publication networks. In: 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 203–212. ACM Press, New York (2010)
15. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: 18th International Conference on Machine Learning, pp. 282–289. Morgan Kaufmann, San Francisco (2001)
16. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In: Neural Information Processing Systems 2001, pp. 841–848. The MIT Press, Cambridge (2002)
17. Fortunato, S.: Community detection in graphs. Physics Reports 486(3-5), 75–174 (2010)
18. Kindermann, R., Snell, J.L.: Markov Random Fields and Their Applications. American Mathematical Society, Providence (1980)
19. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
20. Murphy, K.P., Weiss, Y., Jordan, M.I.: Loopy belief propagation for approximate inference: an empirical study. In: 15th Conference on Uncertainty in Artificial Intelligence, pp. 485–492. Morgan Kaufmann, San Francisco (1999)
21. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. PNAS 105(4), 1118–1123 (2008)
22. Lee, C., Reid, F., McDaid, A., Hurley, N.: Detecting highly overlapping community structure by greedy clique expansion. In: KDD 2010 Workshop on Social Network Mining and Analysis (2010)
23. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. Physical Review E 80(5), 056117 (2009)

Correcting Bias in Statistical Tests for Network Classifier Evaluation

Tao Wang[1], Jennifer Neville[2], Brian Gallagher[3], and Tina Eliassi-Rad[4]

[1] Department of Computer Science,
Purdue University, West Lafayette, IN, USA
[2] Department of Computer Science and Statistics,
Purdue University, West Lafayette, IN, USA
[3] Lawrence Livermore National Laboratory, Livermore, CA, USA
[4] Department of Computer Science,
Rutgers University, Piscataway, NJ, USA

Abstract. It is difficult to directly apply conventional significance tests to compare the performance of network classification models because network data instances are not independent and identically distributed. Recent work [6] has shown that paired t-tests applied to overlapping network samples will result in unacceptably high levels (e.g., up to 50%) of Type I error (i.e., the tests lead to incorrect conclusions that models are different, when they are not). Thus, we need new strategies to accurately evaluate network classifiers. In this paper, we analyze the sources of bias (e.g. dependencies among network data instances) theoretically and propose analytical corrections to standard significance tests to reduce the Type I error rate to more acceptable levels, while maintaining reasonable levels of statistical power to detect true performance differences. We validate the effectiveness of the proposed corrections empirically on both synthetic and real networks.

Keywords: Social network analysis, Network classification.

1 Introduction

A central methodological issue in machine learning research is to compare the empirical performance of two learning algorithms and assess the *significance* of observed performance differences. Generally, to compare two classification algorithms, the available data is repeatedly partitioned (i.e., sampled) into disjoint training and test sets (e.g., using cross-validation). Then the algorithms are used to (1) learn a model from each training set, and (2) apply the learned models to the appropriate test set for prediction. Evaluation of the test set predictions (e.g., using accuracy) results in a *set* of performance measurements, one for each training/test split, for each algorithm. A hypothesis test is often used to assess whether the set of observed scores (for each of the two algorithms) are significantly different—by comparing them to the distribution of scores that would be expected if both sets were drawn from the same underlying distribution (i.e., the null hypothesis that the algorithms perform equivalently).

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 506–521, 2011.

Past work on methodology for accurate algorithm evaluation has mainly focused on data with independent and identically distributed (i.i.d.) instances. Dietterich [2] showed that some statistical tests, in widespread use at the time, had a high probability of Type I error due to sampling procedures that resulted in dependencies among test sets (i.e., they are likely to conclude a significant difference between algorithms when there is none). Owen [7] observed that dependencies among the hypothesis tests greatly affect the variance of the number of false discoveries in which a true null hypothesis was rejected. Other work has shown that the choice of training/test sets can lead to underestimation of variance in the cross-validation estimator of the generalization error [5,1].

However, standard approaches to algorithm evaluation become more challenging in relational learning where the data instances are not independent. In particular, two characteristics of relational learning and collective classification [8] can complicate the application of conventional statistical tests for comparing classification performance: (1) dependence between related instances leads to correlated errors and (2) network structure results in dependence between training and test set samples, which leads to correlated test sets.

Recently Neville et. al [6] conducted an empirical investigation of evaluation bias when learning from non-i.i.d. observations and proposed a novel sampling method called *network cross-validation* (NCV) that can correct for elevated levels of Type I error in network data—but at the expense of decreased statistical power (i.e., legitimate performance differences may not be detected as significant). Note that if a statistical test has biased levels of Type I error, that means many algorithms which appear to be "significantly different" may in fact have equivalent performance; if a statistical test has low statistical power, that means legitimate performance differences between algorithms may not be detected as significant.

In this paper, we consider the problem of *within-network* relational learning, where there are dependencies among data instances and the goal is transductive network learning—models are learned on a partially labeled network and then applied to *collectively* predict the class labels in the remainder of the network (i.e., the unlabeled portion). Within this setting, we demonstrate how the aforementioned network data characteristics contribute to increased Type I error in conventional statistical tests. Our analysis shows that both error correlation and overlapping samples lead to misestimation of the variance that is used in statistical tests. Based on our analysis, we propose an analytical correction to the observed variance which can be used to adjust for the bias and reduce Type I error rates, while maintaining reasonable statistical power. We demonstrate the effectiveness of the correction on both synthetic and real world data, with simulated and real classifiers. Although we evaluate the properties of the corrected significance tests for within-network classification, the findings are also applicable to other learning tasks, where evaluation is conducted with overlapping samples.

2 Network Classifier Evaluation

When comparing the empirical performance of machine learning algorithms, there are two primary methodological decisions: First, the *sampling procedure* dictates how the available data is partitioned into training and test sets for estimation of algorithm performance. Second, the *significance test* takes a set of performance measurements (e.g., accuracy) from the various sampling trials and makes a determination as to whether observed differences reflect a true difference in classifier performance or whether it is likely to have occurred by chance alone.

Sampling procedures: Given a *single*, fully labeled network S of size m, we consider two sampling procedures to generate training (labeled S_L) and test (unlabeled S_U) sets to evaluate within-network classification algorithms.

The first method is *random resampling* (RS). It involves random draws *without replacement* from the sample population (i.e., S) to generate a training/test split ($S_L \cup S_U = S; S_L \cap S_U = \emptyset$). To produce multiple training/test splits, the method samples repeatedly from the single network S, which results in overlapping test sets (i.e., $|S_{U_i} \cap S_{U_j}| \geq 0$). This method has been used extensively in past work on relational learning algorithms (see the survey in [6] for more detail).

The second method is NCV, a new sampling approach proposed by [6]. NCV samples for k disjoint test sets that will be used for *evaluation* ($S_{U_1} \cup ... \cup S_{U_k} = S; S_{U_1} \cap ... \cap S_{U_k} = \emptyset$). For each test set, the training set is selected from the complement of the network (i.e., $S_{L_i} \subseteq S - S_{U_i}$). When the target training set size is less than the size of the complement, this will leave a set of unlabeled nodes that are neither in the test set nor the training set. Since these unlabeled instances will likely be connected to nodes in the test set, collective inference is run over the full set of unlabeled nodes (i.e., $S - S_{L_i}$), but model performance is only evaluated on the nodes assigned to the test set (S_{U_i}). Since NCV only *evaluates* model performance using disjoint test set instances, it eliminates much of the dependency due to overlapping test sets and will not suffer the same level of bias as RS [6].

Significance tests: In within-network learning, after a sampling procedure has been chosen to create training/test splits within a network, models are learned from each training set and the learned models are applied for collective inference over the appropriate test set (i.e., unlabeled portion of the network). The predictions on the test set nodes are evaluated to estimate algorithm performance (e.g., accuracy). This results in a set of performance measurements, one for each training/test split, for each algorithm. A significance test is then used to determine whether the observed performance differences are *significantly* different than would be expected if the performance measures were drawn from the same underlying distribution (i.e., the null hypothesis H_0 : the algorithms perform equivalently).

In this work, we considered both paired and unpaired t-tests for assessments of significance. We are interested in two characteristics of these tests: (1) *Type I error*: the probability of rejecting a *true* null hypothesis, and (2) *Power*: the probability of rejecting a *false* null hypothesis (i.e., 1-Type II error). Ideally the Type I error of a significance test is equal to the chosen significance level α. If a statistical test has biased levels of Type I error (i.e., greater than the significance level α), that implies that many of the conclusions drawn from the test may be incorrect (e.g., algorithms that appear to be different may in fact have equivalent performance). In contrast, if a statistical test has low statistical power, that implies that legitimate performance differences may not be detected as significant.

3 Theoretical Analysis

Here we show theoretically how error correlation and random sampling (i.e., without replacement) from a network affects the variance of average network classification error. To do this, we model the node-level classification errors as Bernoulli random variables and analytically calculate the mean and variance of the average error over repeated samples from the same network. Specifically:

- The input population is a set of m random variables X (i.e., network size=m).
- The population consists of two types of random variables. There are pm random variables of type 1 (i.e., likely errors), which are Bernoulli distributed: $X_i^1 \sim$Bernoulli(q). There are $(1-p)m$ instances of type 0 (i.e., likely correct), which again are Bernoulli distributed: $X_i^0 \sim$Bernoulli($\frac{p}{(1-p)}(1-q)$).
- In the population, there are $|L|$ pairs of "linked" random variables that are correlated. Let ρ be the average correlation between the linked pairs $((X_i, X_j) \in L)$, otherwise we assume that the X_i are independent.
- We sample n random variables $\{X_i\}_{i=1}^n$ without replacement from the population. Since the sampling is without replacement, the random variables X_is are not independent.
- Let $Z_k = \frac{1}{n}\sum_{i=1}^n X_i$ be the average value of the r.v.'s in sample k. We are then interested in the mean and variance of the random variable Z_k, as this corresponds to the estimated error rate of algorithms that is used in statistical tests.

We note that this setup makes two primary assumptions in order to simplify the subsequent analysis. First, we assume that the variance in classification errors throughout the network, across multiple samples, can be represented by the two types of Bernoulli random variables described above. We designed the parameters of the Bernoulli variables to keep the expected value of Z_k equal to p (i.e., the average error), while allowing individual variation of the random variables across multiple samples: $E(Z_k) = E\left(\frac{1}{n}\sum_{i=1}^n X_i\right) = E\left(pX_i^1 + (1-p)X_i^0\right) = pq + (1-p)\frac{p}{(1-p)}(1-q) = p$. Note that if $q = 1$, then the random variables have exactly the same values across all samples (if selected) so this would correspond to sampling from a hypergeometric distribution with pm 1s.

Second, we consider a limited correlation structure in the above model. In particular, we assume (1) uniform correlation among all the linked nodes, and (2) independence among all unlinked nodes in the network. This is a first approximation of the assumptions typical in relational classification models, where the parameters of directly linked nodes are tied and unlinked nodes are considered conditionally independent.

Since we have assumed independence among unlinked nodes, rather than conditional independence, the validity of our proposed model depends on whether the specified covariance matrix is positive definite. Let σ_i be the standard deviation of X_i, then the entries of the covariance matrix will be:

$$Cov(X_i, X_j) = \begin{cases} \sigma_i^2 & i = j \\ \rho \cdot \sigma_i \sigma_j & (X_i, X_j) \in L \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

In the appendix, we specify the conditions under which this matrix will be positive definite, and thus a valid covariance matrix. In practice, we find that even when the matrix is not positive definite, it is reasonable to use for the purposes of correcting evaluation bias.

Given this setup, we can now show the effect of correlation and sampling without replacement on the variance of Z_k. We state the theorems and their interpretations below and include the proofs in the appendix.

Theorem 1. ***Correlated variables increase the variance of*** Z_k

Let $\mathbf{X}$ *be a population of Bernoulli(p) random variables. Assume that a sample of* n *variables are drawn randomly from the population. Let* ρ *be the average correlation between the* X_i *that are "linked", where the probability of linkage is* $\frac{|L|}{n(n-1)}$[1]*, and assume that otherwise the* X_i *are independent. Then the variance of* Z_k *is:*

$$Var(Z_k) = \frac{1}{n} p(1-p) \left[1 + \rho \frac{|L|}{n} \right] \tag{2}$$

We refer to this variance of the average error, when there is error correlation, as $Var_{corr}(Z_k)$. Note that, other than for very specific graph structures (e.g., bipartite graphs), if relational data are correlated, autocorrelation is positive and ρ will be greater than zero. Thus, as ρ or $|L|$ (i.e., number of correlated pairs) increase, $Var_{corr}(Z_k)$ also increases.

Theorem 2. ***Sampling without replacement decreases the variance of*** Z_k

Let $\mathbf{X}$ *be a population of* m *Bernoulli random variables as described above, with* pm X^1 *variables (i.e., type 1) and* $(1-p)m$ X^0 *variables (i.e., type 0), where all the* X_i *are independent. Assume that a sample of* n *variables are drawn randomly from the population. Then the variance of* Z_k *is:*

$$Var(Z_k) = \frac{1}{n} p(1-p) \left[1 - \frac{(n-1)}{(m-1)} \left(\frac{q-p}{1-p} \right)^2 \right] \tag{3}$$

[1] Note that $n(n-1)$ is the number of possible directed edges in a network of n nodes.

We refer to this variance of the average error, when there is overlap between samples due to resampling, as $Var_{rs}(Z_k)$. Note that when $q = p$, the variables correspond to independent Bernoullis across samples and the overall variance reduces to the case when each sample is independent: $Var(Z_k) = \frac{1}{n}p(1-p)$. When $q = 1$, the random variables have exactly the same value across different samples and the variance corresponds to sampling from a Hypergeometric distribution: $Var(Z_k) = \frac{1}{n}p(1-p)\left[\frac{m-n}{m-1}\right]$.

We can now extend the results of Theorem 2, to show the joint effect of correlation and sampling without replacement on the variance of Z_k.

Theorem 3. ***Variance of Z_k with variable correlation and sampling without replacement***

Let $\mathbf{X}$ be a population of m Bernoulli random variables as described above, with pm X^1 variables (i.e., type 1) and $(1-p)m$ X^0 variables (i.e., type 0). Let ρ be the average correlation between the X_i that are "linked", where the probability of linkage is $\frac{|L|}{n(n-1)}$, and assume otherwise the X_i are independent. Assume a sample of n variables are drawn randomly from the population. Let $c = \sqrt{1-2p+pq}$. Then the variance of Z_k is:

$$Var(Z_k) = \frac{1}{n}p(1-p)\left[1 - \frac{(n-1)}{(m-1)}\left(\frac{q-p}{1-p}\right)^2 + \frac{|L|\rho}{n(m-1)}\left(\frac{1-q}{1-p}\right)\left[pmq - q + 2mc\sqrt{pq} + mc^2 - \frac{c^2}{(1-p)}\right]\right] \quad (4)$$

We refer to this variance of the average error, when there is both overlap between samples and error correlation, as $Var_{obs}(Z_k)$. This is the variance that is observed in networks domains when random sampling is used. Finally, we can use these results to show these two effects combine together to bias conventional statistical tests for network domains.

Theorem 4. ***Sampling without replacement and error correlation increase Type I error***

Let algorithm A and algorithm B have equal error rates of p on network datasets drawn from the same domain D. Let X_i be the classification error for node i and assume that $X_{i.A}$ and $X_{i.B}$ (the error made by algorithm A and B respectively) are Bernoulli distributed as described above, i.e., with probability p, $X_{i.A/B}$ is of type 1 and with probability $(1-p)$, $X_{i.A/B}$ is of type 0. Let ρ be the average correlation between the X_i, X_j that are linked (i.e., $e_{ij} \in L$) and assume that otherwise the X_i are independent. Assume that k test sets, each of size n, are drawn from the network of m nodes.

Let $\mathbf{Z^A} = \{Z_1^A, Z_2^A, \ldots, Z_k^A\}$ and $\mathbf{Z^B} = \{Z_1^B, Z_2^B, \ldots, Z_k^B\}$ be the set of average test set errors ($Z_j = \frac{1}{n}\sum_i X_i$) for test set $j = [1, k]$. Let $c = \sqrt{1-2p+pq}$. Then an unpaired t-test over $\mathbf{Z^A}$ and $\mathbf{Z^B}$ will underestimate the variance of the *null distribution by:* $\mathit{\Delta} = \frac{1}{n}p(1-p)\left[\frac{(n-1)}{(m-1)}\left(\frac{q-p}{1-p}\right)^2 + \rho\frac{|L|}{n}\left[1 - \frac{1}{(m-1)}\left(\frac{1-q}{1-p}\right)\right.\right.$ $\left.\left.\left[pmq - q + 2mc\sqrt{pq} + mc^2 - \frac{c^2}{(1-p)}\right]\right]\right]$.

As ρ (the amount of error correlation) or q (the correlation of node error across samples) increases, the amount of underestimation (i.e., $\mathit{\Delta}$) increases. This increases the probability of a Type I error in the following way. For unpaired tests, the t-statistic is: $\hat{t} = \frac{\bar{Z^A} - \bar{Z^B}}{\sqrt{Var(Z_{A/B})}\cdot\sqrt{\frac{2}{k}}}$. where $\bar{Z^A} = \frac{1}{k}\sum_j Z_j^A$ is the average of $Z_j s$ in $\mathbf{Z^A}$ (averaging the average test set errors made by algorithm A over k tet sets) , $\bar{Z^B} = \frac{1}{k}\sum_j Z_j^B$ is the average of $Z_j s$ in $\mathbf{Z^B}$, and $Var(Z_{A/B})$ is the pooled sample variance. Since $Var_{obs}(Z_k) < Var_{corr}(Z_k)$, the result will be that $\hat{t}_{obs} > \hat{t}_{corr}$ and thus $P(\hat{t}_{obs}|T) < P(\hat{t}_{corr}|T)$, where T is the appropriate t distribution with $dof = 2k - 2$. Thus using $Var_{obs}(Z_k)$ instead of $Var_{corr}(Z_k)$, it is more likely that the null hypothesis will be rejected even when it holds, and as such Type I error will increase. This effect will impact paired t-tests in a similar way, as the decrease in observed variance of Z_j^A and Z_j^B will also result in an underestimate of the *difference variance* $Var(Z_j^A - Z_j^B)$, which is used instead of the pooled sample variance.

4 Analytical Correction for Bias

Based on the theoretical analysis in Section 3, we propose an analytical adjustment to correct for the bias due to repeated sampling without replacement. We would like to remove the effects of resampling, and adjust the observed variance $Var_{obs}(Z_k)$ to make it equal to the variance we would expect just due to correlation: $Var_{corr}(Z_k) = \frac{1}{n}p(1-p)[1 + \rho\frac{|L|}{n}]$. To achieve this, we simply add in the correction factor $\mathit{\Delta}$ from Theorem 4 above: $Var_{new}(Z_k) = Var_{obs}(Z_k) + \mathit{\Delta} = Var_{corr}(Z_k)$.

Correction for unpaired t-test: The correction can be used in an unpaired t-test in the following manner. We estimate model error (for each model) in the conventional manner, recording average performance over multiple test sets. After computing the variance of the average performances for a particular model (i.e., $Var_{obs}(Z_k)$), we compute the appropriate $\mathit{\Delta}$ from above and use it to scale the observed variance. Then the corrected variance $Var_{new}(Z_k)$ is used in place of the observed variance in the standard formulation of the unpaired t-test.

Correction for paired t-test: For the paired t-test, we can use the correction to rescale each observed value before computing the differences and variance. The idea is to compute the standardized value with the original variance (Var_{obs}) and then *unstandardize* using the corrected variance (Var_{new}). Let x^A be an observed error value for algorithm A. Let μ^A be the mean (observed) error for algorithm A. Let $\sigma^A_{obs} = (Var^A_{obs})^{\frac{1}{2}}$ be the observed standard deviation of the average performance of algorithm A. Let $\sigma^A_{new} = (Var^A_{new})^{\frac{1}{2}}$ be the corrected standard deviation of algorithm A. Then the adjustment for each measured performance value x^A is the following: $x_c^A = \left[\left(\frac{x^A-\mu^A}{\sigma_{obs}^A}\right)\cdot\sigma_{new}^A\right]+\mu^A = \left(\frac{\sigma_{new}^A}{\sigma_{obs}^A}\right)x^A+ \left(1-\frac{\sigma_{new}^A}{\sigma_{obs}^A}\right)\mu^A$ The same adjustment is then applied to errors for algorithm B, with appropriate mean and variances. Once all the observed errors are adjusted, we can then compute the paired t-test in the standard way.

The correction Δ requires values for the parameters: $n, m, p, q, \rho, |L|$. We can easily calculate $n, m, |L|$ from the properties of the training/test networks used in a particular evaluation. Also, p, q, ρ can be estimated from the training/test network evaluations. For the experiments below, we use the average misclassification over all instances in a test set for p, the average misclassification for each instance across multiple test sets for q, and for ρ we use the ϕ coefficient to measure the correlation of errors for linked instances in the network (i.e., calculate ϕ coefficient from a contingency table that shows the association of prediction errors of a pair of linked instances). In the following sections we report results for paired tests only. Experiments with unpaired tests yielded qualitatively similar results.

5 Experimental Results

To investigate the effectiveness of our proposed correction with random resampling (RS-C), for significance tests of network classifiers, we conducted experiments with both simulated and real relational classifiers under varying data characteristics, using synthetic data and data from the Internet Movie Database (imdb.com).

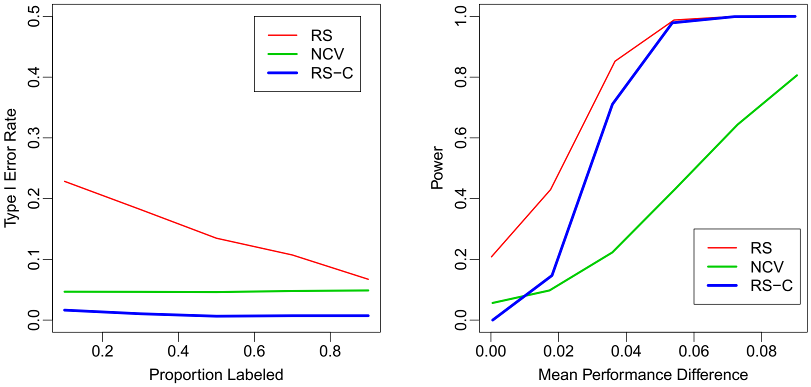
We compare the Type I error rates and statistical power of RS, NCV, and RS-C using paired t-tests. In all the experiments, both Type I error rates and statistical power rates were averaged over 500 (simulated) or 50 (synthetic/real) trials. For a given dataset, in each trial we *sample* from the network, either using random sampling (RS) or using network cross-validation (NCV), to create 10 training/test splits (subnetworks). Then we learn classifiers (using two competing algorithms A and B) on the training subnetwork and apply the learned classifiers on its corresponding test subnetwork to measure its performance (e.g. average error rate). To compare performance, we conducted significant tests ($\alpha = 0.05$) to either accept or reject the null hypothesis that the performance of algorithm A and B are equivalent. When the experiments are designed so that two learned classifiers have equivalent error rates, any rejection of the null hypothesis corresponds to a Type I error (i.e., false positive identification of a difference when it does not exist). However, when the two classifiers perform differently, a rejection of the null hypothesis represents the *statistical power* of the test (i.e., true positive identification of a difference when it exists). We calculate and report the proportion of trials for which the null hypothesis was rejected (i.e. Type I error or power in its corresponding experimental setup).

5.1 Experiments with Simulated Classifiers

Here we replicate the experiments of [6] to analyze test characteristics with simulated classifiers. We simulate the correlated errors observed in real network classifiers by dividing data instances into disjoint groups and assigning "classification errors" such that errors are correlated among the instances within a group. We simulate two group-based classifiers A and B, ensuring that A and B have the same error rate (p) while still making different kinds of errors (i.e., A misclassifies different groups from B). Each trial utilizes datasets with default parameters $m = 300$, $p = 0.1$, and $q = 0.9$.

Figure 1(a) shows the effects of varying the proportion of labeled data for training. In these experiments, algorithms A and B have equal error rates of $p = 0.1$ so rejecting the null hypothesis corresponds to a Type I error. For RS, the Type I error rate increases as *propLabeled* decreases. This result is expected since the degree of overlap between test sets increases as the number of unlabeled instances increases. Since NCV disallows overlapping test sets by design, it is not susceptible to this problem, achieving uniformly low Type I error rates. The corrected test, RS-C, exhibits a further reduction in type I error over NCV since it accounts for error correlation as well as test set overlap.

Figure 1(b) shows the statistical power of the tests when the difference in error rates between A and B is varied ($propLabeled = 0.3$). In this case, since the algorithm error rates are different, a rejection of the null hypothesis corresponds to a true positive. RS has the highest statistical power overall, but when its high Type I error rates are taken into account, RS has little practical utility. RS-C, on the other hand, is able to maintain low Type I error while achieving a reasonable amount of statistical power. For example when there is a 4% difference in error rates, RS-C will be able to detect it 80% of the time. NCV has substantially lower statistical power—it will only be able to detect a 4% difference 20% of the time.



(a) Type I error.

(b) Statistical power.

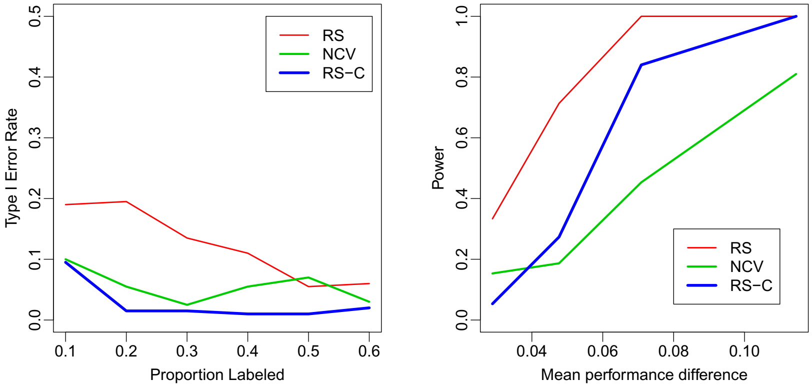
Fig. 1. Type I error and power experiments on synthetic data with simulated classifiers. (Left) Type I error as proportion of labeled data increases. (Right) Statistical power as the difference between classifiers increases.

5.2 Experiments with Real Classifiers

To further investigate RS-C, we compare the collective classification models used in [6]: weighted-vote relational neighbor (wvRN) [4] and network-only Bayes classifier (nBC) [4]. For both models, we use relaxation labeling for collective inference. To estimate Type I error, we handicap the *better* performing model (wvRN) until the performance difference between the models is negligible (i.e., ≤ 0.005). This is achieved by randomly selecting $b\%$ of the wvRN's predictions and perturbing those probabilities toward the opposite class. We searched for a value of b that resulted in a performance difference of ≤ 0.005 between the two models on a separate set of *calibration* networks. To estimate statistical power, we handicap the *worse* performing model (nBC) to increase the performance difference between the two models. We used $b = [0.025, 0.075, 0.15, 0.3]$ and measured the resulting performance difference, which is reported in Figure 2(b) and 3(b).

Results on synthetic data: In this set of experiments, we use synthetic datasets as described in [6]. The generated networks have size $m = 300$ with average autocorrelation$= 0.40$ and class prior $P(+) = 0.70$. The data is designed so that wvRN and nBC will make classification errors on *different* nodes. To measure Type I error rates and power of the statistical tests, we used four synthetic networks (in addition a set of 50 calibration networks).

Figure 2(a) plots the Type I error rates for three statistical tests. Notably, the level of Type I error exhibited by RS-C is significantly lower than that of RS (> 50% reduction in error). RS-C Type I error is also slightly lower than that of NCV. Figure 2(b) plots the power of each statistical test on networks with 30% labeled nodes. Here we observe, that RS-C again achieves much higher power than NCV. This is due to its use of larger test sets sizes—after correcting



(a) Type I error.

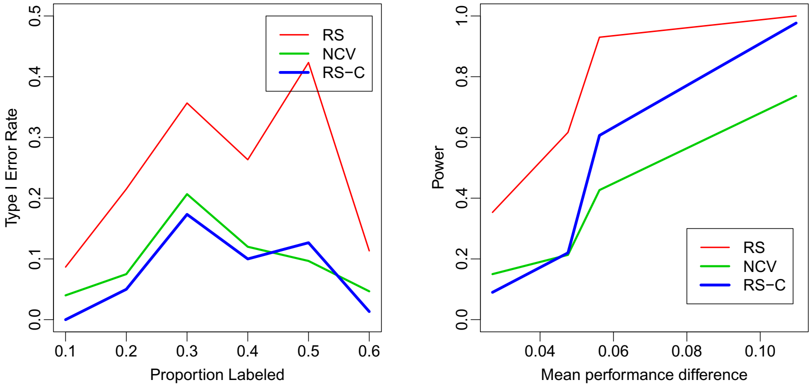
(b) Statistical power.

Fig. 2. Type I error and power experiments on synthetic data with real classifiers. (Left) Type I error as proportion of labeled data increases. (Right) Statistical power as the difference between classifiers increases.

for overlap, the *effective* sample size is still larger than the disjoint sets used in NCV. For example, on a network of 300 nodes with 30% labeled nodes, RS-C uses test set sample of 210 nodes while NCV only use a test set of 30 nodes (because of 10 cross validation). Note that the test set of 210 nodes in RS-C are not independent sample. The overlap correction will adjust its sample size downward, but the effective sample size of RS-C will still be larger than 30.

Results on real data: In the second set of experiments, we use data from the Internet Movie Database (IMDB). We collected a sample of 1,543 movies released in the United States between 2003 and 2007, with their associated producers and studios. We create six disjoint network samples using stratified sampling by studios. Within each partition, we created links among movies with a common producer. The resulting networks have an average size of 257 nodes and the movies have average degree of 16. The classification task is to predict whether the movie will make more than $60$mil$ in total box office receipts. The average autocorrelation in these networks is 0.35.

Figure 3(a) and 3(b) show the Type I error and statistical power for each test respectively. The relative performance of the statistical tests is similar across the synthetic data and the real network data. RS-C has Type I error rates comparable to NCV and significantly lower than RS. Again RS-C has much higher power than NCV for detecting the algorithm differences in real network data.



(a) Type I error.

(b) Statistical power.

Fig. 3. Type I error and power experiments on real data (IMDB) with real classifiers. (Left) Type I error as proportion of labeled data increases. (Right) Statistical power as the difference between classifiers increases.

6 Conclusion

We investigated two biases present in statistical tests for within-network classification algorithms: (1) correlated errors among related instances and (2) overlap between samples. These biases increase the Type I error to unacceptably high-levels. To adjust for these biases, we developed analytical corrections to the empirical estimates of variance. Experiments on real and synthetic data, using real and simulated classifiers demonstrate that our corrections reduce the Type I error while maintaining good statistical power. Compared to the network cross-validation, our corrections result in a significant increase in statistical power.

Acknowledgments. We thank Yao Zhu and S.V.N. Vishwanathan for helpful discussions. This research was supported by NSF Science & Technology Center grant CCF-0939370 and NSF under contract number SES-0823313. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-CONF-485754).

References

1. Bengio, Y., Grandvalet, Y.: No unbiased estimator of the variance of k-fold cross-validation. Journal of Machine Learning Research 5, 1089–1105 (2004)
2. Dietterich, T.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10, 1895–1923 (1998)
3. Franklin, J.N.: Matrix Theory. Dover Publications, Mineola (1993)
4. Macskassy, S., Provost, F.: Classification in networked data: A toolkit and a univariate case study. Journal of Machine Learning Research 8, 935–983 (2007)
5. Nadeau, C., Bengio, Y.: Inference for the generalization error. Machine Learning Journal 52(3), 239–281 (2003)
6. Neville, J., Gallagher, B., Eliassi-Rad, T., Wang, T.: Correcting evaluation bias of relational classifiers with network cross validation. Knowledge and Information Systems, 1–25 (2011)
7. Owen, A.B.: Variance of the number of false discoveries. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67, 411–426 (2005)
8. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 29(3), 93–106 (2008)

Appendix

Conditions for Covariance Matrix Validity

The covariance matrix, denoted as $\mathbf{\Sigma}$, can be specified in matrix form as:

$$\mathbf{\Sigma} := \rho(\sigma\sigma^T). * \mathbf{A} + \mathrm{diag}(\sigma. * \sigma) \tag{5}$$

where $\sigma = [\sigma_1, \ldots, \sigma_i, \ldots, \sigma_n]$, $\mathbf{A}_{ij} = 1$ if instance i and j are linked and 0 otherwise, diag has the usual semantics, and $.*$ is the pointwise product.

To show the conditions under which the specified covariance matrix is valid, it is enough to show when $\mathbf{\Sigma}$ is positive definite.

Lemma 1. *Let $\nu_{\min}$ denote the minimum eigenvalue of matrix $\mathbf{H} = (\sigma\sigma^T). * \mathbf{A}$, and $\psi_{\min}$ denote the minimum eigenvalue of matrix $\mathbf{P} = \mathrm{diag}(\sigma. * \sigma)$. If ρ satisfies:*

$$\rho \begin{cases} < -\frac{\psi_{\min}}{\nu_{\min}} & \textit{if } \nu_{\min} > 0 \\ > -\frac{\psi_{\min}}{\nu_{\min}} & \textit{if } \nu_{\min} < 0, \end{cases} \tag{6}$$

then the covariance matrix $\mathbf{\Sigma}$ defined above is positive definite.

Proof. To ensure that $\boldsymbol{\Sigma}$ *is positive definite it is sufficient to show that* $\lambda_{\min} > 0$*, where* $\lambda_{\min}$ *denotes the minimum eigenvalue of* $\boldsymbol{\Sigma}$*. By Weyl's inequality [3] we have* $\rho\nu_{\min} + \psi_{\min} \leq \lambda_{\min}$*, from which it directly follows that* $\lambda_{\min} > 0$ *whenever (6) is satisfied.*

Even though Lemma 1 gives admissible values of ρ to ensure that the covariance matrix is positive definite, we observe empirically that other values of ρ also yield good analytical corrections in practice. In other words, even if the covariance matrix underlying the correction is not positive definite, our adjustment method is still able to correct for the evaluation bias and correctly assess significant algorithm differences.

Proof of Theorem 1

Proof.

$$\begin{aligned}Var(Z_k) &= Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) && (7)\\ &= \frac{1}{n^2}\left(\sum_{i=1}^{n} Var(X_i) + \sum_{i=1}^{n}\sum_{j\neq i}^{n} Cov(X_i, X_j)\right) && (8)\\ &= \frac{1}{n^2}\left(n \cdot p(1-p) + |L|\rho \cdot p(1-p)\right) && (9)\\ &= \frac{1}{n}p(1-p)\left[1 + \rho\frac{|L|}{n}\right] && (10)\end{aligned}$$

Proof of Theorem 2

Proof. First we consider the joint probability of two instances, based on sampling without replacement:

$$\begin{aligned}&P(X_i\!=\!1\!\wedge\! X_j\!=\!1)\\ &= P(X_i \in X^1 \wedge X_i = 1)P(X_j \in X^1 \wedge X_j = 1|X_i \in X^1)+\\ &\quad P(X_i \in X^1 \wedge X_i = 1)P(X_j \in X^0 \wedge X_j = 1|X_i \in X^1)+\\ &\quad P(X_i \in X^0 \wedge X_i = 1)P(X_j \in X^1 \wedge X_j = 1|X_i \in X^0)+\\ &\quad P(X_i \in X^0 \wedge X_i = 1)P(X_j \in X^0 \wedge X_j = 1|X_i \in X^0) && (11)\\ &= \left[\left(\frac{pm}{m}q\right)\left(\frac{pm-1}{m-1}q\right)\right]+\\ &\left[\left(\frac{pm}{m}q\right)\left(\frac{(1-p)m}{m-1}\frac{p}{1-p}(1-q)\right)\right]+\\ &\left[\left(\frac{(1-p)m}{m}\frac{p}{1-p}(1-q)\right)\left(\frac{pm}{m-1}q\right)\right]+\\ &\left[\left(\frac{(1-p)m}{m}\frac{p}{1-p}(1-q)\right)\left(\frac{(1-p)m-1}{m-1}\frac{p}{1-p}(1-q)\right)\right] && (12)\end{aligned}$$

$$= \frac{p}{(m-1)}\left[pm - q^2 - \frac{p(1-q)^2}{(1-p)}\right] \tag{13}$$

Now consider the covariance of two instances, based on sampling without replacement:

$$\begin{aligned}
&Cov(X_i, X_j)\\
&= E(X_iX_j) - E(X_i)E(X_j) &(14)\\
&= P(X_i = 1 \wedge X_j = 1) - p \cdot p &(15)\\
&= \frac{p}{(m-1)}\left[pm - q^2 - \frac{p(1-q)^2}{(1-p)}\right] - p^2 &(16)\\
&= -\frac{p(1-p)}{(m-1)}\left[\frac{(q-p)^2}{(1-p)^2}\right] &(17)
\end{aligned}$$

With the covariance, we can compute the overall variance based on sampling without replacement:

$$\begin{aligned}
Var(Z_k) &= Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) &(18)\\
&= \frac{1}{n^2}\left[\sum_{i=1}^{n} Var\left(X_i\right) + \sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n} Cov(X_i, X_j)\right] &(19)\\
&= \frac{1}{n}\left[p(1-p) - (n-1)\frac{p(1-p)}{(m-1)}\left[\frac{(q-p)^2}{(1-p)^2}\right]\right] &(20)\\
&= \frac{1}{n}p(1-p)\left[1 - \frac{(n-1)}{(m-1)}\left(\frac{q-p}{1-p}\right)^2\right] &(21)
\end{aligned}$$

Proof of Theorem 3

Proof. To combine the covariance based on error correlation with the covariance based on overlap, we need to determine the effect of the correlation on the conditional probability of a linked instance, i.e., $P(X_j = 1|X_i = 1, e_{ij} \in L)$. We can derive this from the relationship between correlation and covariance:

$$\begin{aligned}
Cov(X_i, X_j|e_{ij} \in L) &= Corr(X_i, X_j|e_{ij})Var(X_i)^{\frac{1}{2}}Var(X_j)^{\frac{1}{2}} &(22)\\
E(X_iX_j|e_{ij} \in L) - E(X_i)E(X_j) &= \rho \cdot Var(X_i)^{\frac{1}{2}}Var(X_j)^{\frac{1}{2}} &(23)\\
P(X_j|X_i, e_{ij} \in L) &= E(X_j) + \frac{\rho \cdot Var(X_i)^{\frac{1}{2}}Var(X_j)^{\frac{1}{2}}}{E(X_i)} &(24)
\end{aligned}$$

We can then enumerate the conditional probabilities for each of the four possible worlds for (X_i, X_j):

$$P(X_j^1|X_i^1) = E(X_j^1) + \frac{\rho Var(X_i^1)^{\frac{1}{2}}Var(X_j^1)^{\frac{1}{2}}}{E(X_i^1)} = q + \rho(1-q) \tag{25}$$

$$P(X_j^0|X_i^1) = E(X_j^0) + \frac{\rho Var(X_i^1)^{\frac{1}{2}} Var(X_j^0)^{\frac{1}{2}}}{E(X_i^1)} \tag{26}$$

$$= \frac{p(1-q)}{1-p} + \rho \frac{(1-q)}{(1-p)} \sqrt{\frac{p(1-2p+pq)}{q}} \tag{27}$$

$$P(X_j^1|X_i^0) = E(X_j^1) + \frac{\rho Var(X_i^0)^{\frac{1}{2}} Var(X_j^1)^{\frac{1}{2}}}{E(X_i^0)} \tag{28}$$

$$= q + \rho \sqrt{\frac{q(1-2p+pq)}{p}} \tag{29}$$

$$P(X_j^0|X_i^0) = E(X_j^0) + \frac{\rho Var(X_i^0)^{\frac{1}{2}} Var(X_j^0)^{\frac{1}{2}}}{E(X_i^0)} \tag{30}$$

$$= \frac{p(1-q)}{1-p} + \rho \left(\frac{1-2p+pq}{1-p} \right) \tag{31}$$

Now we can incorporate these conditional probabilities into the calculation of $P(X_i, X_j)$ and $Cov(X_i, X_j)$, incorporating both correlation and sampling without replacement. Let $c = \sqrt{1-2p+pq}$, then:

$$P(X_i{=}1 \wedge X_j{=}1) \tag{32}$$

$$\begin{aligned}
&= \left[\left(\frac{pm}{m} q \right) \left(\frac{pm-1}{m-1} [q + \frac{|L|}{n(n-1)} \rho (1-q)] \right) \right] + \\
&\left[\left(\frac{pm}{m} q \right) \left(\frac{(1-p)m}{m-1} \left[\frac{p}{1-p} (1-q) + \frac{|L|}{n(n-1)} \rho \frac{(1-q)}{(1-p)} c \sqrt{\frac{p}{q}} \right] \right) \right] + \\
&\left[\left(\frac{(1-p)m}{m} \frac{p(1-q)}{1-p} \right) \left(\frac{pm}{m-1} \left[q + \frac{|L|}{n(n-1)} \rho c \sqrt{\frac{q}{p}} \right] \right) \right] + \\
&\left[\left(\frac{(1-p)m}{m} \frac{p(1-q)}{1-p} \right) \left(\frac{(1-p)m-1}{m-1} \left[\frac{p(1-q)}{1-p} + \frac{|L|}{n(n-1)} \rho \left(\frac{c^2}{1-p} \right) \right] \right) \right]
\end{aligned} \tag{33}$$

$$\begin{aligned}
&= \frac{p}{(m-1)} \left[pm - q^2 - \frac{p(1-q)^2}{(1-p)} \right] + \\
&\frac{|L|}{n(n-1)} \left(\frac{pq(1-q)\rho}{m-1} \right) \left[pm - 1 + 2mc\sqrt{\frac{p}{q}} + \frac{mc^2}{q} - \frac{c^2}{q(1-p)} \right]
\end{aligned} \tag{34}$$

$$Cov(X_i, X_j) = E(X_i X_j) - E(X_i)E(X_j) \tag{35}$$

$$= P(X_i = 1 \wedge X_j = 1) - p \cdot p \tag{36}$$

$$\begin{aligned}
&= \frac{p}{(m-1)} \left[pm - q^2 - \frac{p(1-q)^2}{(1-p)} \right] - p^2 + \\
&\frac{|L|}{n(n-1)} \left(\frac{pq(1-q)\rho}{m-1} \right) \left[pm - 1 + 2mc\sqrt{\frac{p}{q}} + \frac{mc^2}{q} - \frac{c^2}{q(1-p)} \right]
\end{aligned} \tag{37}$$

$$= \frac{p(1-p)}{(m-1)} \left[- \left(\frac{q-p}{1-p} \right)^2 + \frac{|L|\rho}{n(n-1)} \left(\frac{1-q}{1-p} [pmq - q + 2mc\sqrt{pq} + mc^2 - \frac{c^2}{(1-p)} \right] \right]$$

Now we can compute the overall variance of Z_k, with correlation as well as sampling without replacement:

$$Var(Z_k) = Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\left[\sum_{i=1}^{n} Var\left(X_i\right) + \sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n} Cov(X_i, X_j)\right] \tag{38}$$

$$= \frac{1}{n}\left[p(1-p) - \frac{n(n-1)}{n}\frac{p(1-p)}{(m-1)}\left[\left(\frac{q-p}{1-p}\right)^2 - \right.\right.$$

$$\left.\left.\frac{|L|\rho}{n(n-1)}\left(\frac{1-q}{1-p}\right)\left[pmq - q + 2mc\sqrt{pq} + mc^2 - \frac{c^2}{(1-p)}\right]\right]\right] \tag{39}$$

$$= \frac{1}{n}p(1-p)\left[1 - \frac{(n-1)}{(m-1)}\left(\frac{q-p}{1-p}\right)^2 + \right.$$

$$\left.\frac{|L|\rho}{n(m-1)}\left(\frac{1-q}{1-p}\right)\left[pmq - q + 2mc\sqrt{pq} + mc^2 - \frac{c^2}{(1-p)}\right]\right] \tag{40}$$

Proof of Theorem 4

Proof. The unpaired t-test uses the average (i.e., *pooled*) variance of Z^A and Z^B for the null distribution. Since the error distribution of A and B are equal, the average is equal to the variance of a single algorithm. When the nodes are repeatedly sampled without replacement, we know from Theorem 3 that the observed variance of Z_k will be the following: $Var_{obs}(Z_k) = \frac{1}{n}p(1-p)\left[1 - \frac{(n-1)}{(m-1)}\left(\frac{q-p}{1-p}\right)^2 + \frac{|L|\rho}{n(m-1)}\left(\frac{1-q}{1-p}\right)\left[pmq - q \; + 2mc\sqrt{pq} + mc^2 - \frac{c^2}{(1-p)}\right]\right.$, where $c = \sqrt{1-2p+pq}$. However, when there is error correlation ρ among the instances in the data, from Theorem 1 we know that the variance of Z_k with independent samples is the following: $Var_{corr}(Z_k) = \frac{1}{n}p(1-p)\left[1 + \rho\frac{|L|}{n}\right]$. Since the t-test assumes independent samples, the variance of the null distribution should correspond to the variance without repeated sampling $Var_{corr}(Z_k)$. If the observed variance $Var_{obs}(Z_k)$ is used in the t-test, it will result in an underestimate of Δ:

$$\Delta = Var_{corr}(Z_k) - Var_{obs}(Z_k) \tag{41}$$

$$= \frac{1}{n}p(1-p)\left[1 + \rho\frac{|L|}{n}\right] - \frac{1}{n}p(1-p)\left[1 - \frac{(n-1)}{(m-1)}\left(\frac{q-p}{1-p}\right)^2\right] +$$

$$\left[\frac{|L|\rho}{n(m-1)}\left(\frac{1-q}{1-p}\right)\left[pmq - q + 2mc\sqrt{pq} + mc^2 - \frac{c^2}{(1-p)}\right]\right] \tag{42}$$

$$= \frac{1}{n}p(1-p)\left[\frac{(n-1)}{(m-1)}\left(\frac{q-p}{1-p}\right)^2 + \right.$$

$$\left.\rho\frac{|L|}{n}\left[1 - \frac{1}{(m-1)}\left(\frac{1-q}{1-p}\right)\left[pmq - q + 2mc\sqrt{pq} + mc^2 - \frac{c^2}{(1-p)}\right]\right]\right] \tag{43}$$

Differentiating Code from Data in x86 Binaries*

Richard Wartell, Yan Zhou, Kevin W. Hamlen,
Murat Kantarcioglu, and Bhavani Thuraisingham

Computer Science Department,
University of Texas at Dallas,
Richardson, TX 75080
{rhw072000,yan.zhou2,hamlen,muratk,bhavani.thuraisingham}@utdallas.edu

Abstract. Robust, static disassembly is an important part of achieving high coverage for many binary code analyses, such as reverse engineering, malware analysis, reference monitor in-lining, and software fault isolation. However, one of the major difficulties current disassemblers face is differentiating code from data when they are interleaved. This paper presents a machine learning-based disassembly algorithm that segments an x86 binary into subsequences of bytes and then classifies each subsequence as code or data. The algorithm builds a language model from a set of pre-tagged binaries using a statistical data compression technique. It sequentially scans a new binary executable and sets a breaking point at each potential code-to-code and code-to-data/data-to-code transition. The classification of each segment as code or data is based on the minimum cross-entropy. Experimental results are presented to demonstrate the effectiveness of the algorithm.

Keywords: statistical data compression, segmentation, classification, x86 binary disassembly.

1 Introduction

Disassemblers transform machine code into human-readable assembly code. For some x86 executables, this can be a daunting task in practice. Unlike Java bytecode and RISC binary formats, which separate code and data into separate sections or use fixed-length instruction encodings, x86 permits interleaving of code and static data within a section and uses variable-length, unaligned instruction encodings. This trades simplicity for brevity and speed, since more common instructions can be assigned shorter encodings by architecture designers. An unfortunate consequence, however, is that hidden instructions can be concealed within x86 binaries by including jump instructions that target the interior of another instruction's encoding, or that target bytes that resemble

* This material is based upon work supported by the AFOSR under contracts FA9550-08-1-0044, FA9550-10-1-0088, and FA9550-08-1-0265; by the NIH through grant number 1R01LM009989; and by the NSF through grant numbers Career-0845803, CNS-0964350, and CNS-1016343.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 522–536, 2011.

data. This causes these bytes to be interpreted as code at runtime, executing code that does not appear in the disassembly. Malicious code is therefore much easier to conceal in x86 binaries than in other formats. To detect and identify potential attacks or vulnerabilities in software programs, it is important to have a comprehensive disassembly for analyzing and debugging the executable code.

In software development contexts, robust disassembly is generally achieved by appealing to binary debugging information (e.g., symbol/relocation tables) that is generated by most compilers during the compilation process. However, such information is typically withheld from consumers of proprietary software in order to discourage reverse engineering and to protect intellectual property. Thus, debugging information is not available for the vast majority of COTS binaries and other untrusted mobile code to which reverse engineering is typically applied.

Modern disassemblers for x86 binaries therefore employ a variety of heuristic techniques to accurately differentiate bytes that comprise instructions from those that comprise static data. The techniques are heuristic because fully correct x86 disassembly is provably undecidable: Bytes are code if and only if they are reachable at runtime—a decision that reduces to the halting problem.

IDA Pro [9] is widely acknowledged as the best x86 static disassembly tool currently available for distinguishing code from data in arbitrary binaries (cf., [1,6,12]). It combines straight-line, heuristic, and execution emulation-based disassembly while also providing an extensive GUI interface and multiple powerful APIs for interacting with the disassembly data. Recent work has applied model-checking and abstract interpretation to improve upon IDA Pro's analysis [12,13], but application of these technologies is currently limited to relatively small binaries, such as device drivers, for which these aggressive analyses remain tractable. All other widely available disassemblers to our knowledge take a comparatively simplistic approach that relies mainly upon straight-line disassembly, and that therefore requires the user to manually separate code from data during binary analysis. Our tests therefore focus on comparing the accuracy of our algorithm to that of IDA Pro.

Disassembly heuristics employed by IDA Pro include the following:

- *Code entry point.* The starting point for analyzing an executable is the address listed in the header as the code entry point. That address must hold an instruction, and will hopefully lead to successfully analyzing a large portion of the executable.
- *Function prologues and epilogues.* Many function bodies compiled by mainstream compilers begin with a recognizable sequence of instructions that implement one of the standard x86 calling conventions. These byte sequences are assumed by IDA Pro to be the beginnings of reachable code blocks.
- *Direct jumps and calls.* The destination address operand of any static jump instruction that has already been classified as reachable code is also classified as reachable code.
- *Unconditional jumps and returns.* Bytes immediately following a reachable, unconditional jump or return instruction are considered as potential data

bytes. These often contain static data such as jump tables, padding bytes, or strings.

However, despite a decade of development and tuning, IDA Pro nevertheless fails to reliably distinguish code from data even in many non-malicious, non-obfuscated x86 binaries. Some common mistakes include the following:

- *Misclassifying data as returns.* IDA Pro frequently misclassifies isolated data bytes within data blocks as return instructions. Return instructions have a one-byte x86 encoding and are potential targets of computed jumps whose destinations are not statically decidable. This makes them extremely difficult to distinguish from data. IDA Pro therefore often misidentifies data bytes that happen to match the encoding of a return instruction.
- *16-bit legacy instructions.* The x86 instruction set supports legacy 16-bit addressing modes, mainly for reasons of backward compatibility. The vast majority of genuinely reachable instructions in modern binaries are 32- or 64-bit. However, many data bytes or misaligned code bytes can be misinterpreted as 16-bit instructions, leading to flawed disassemblies.
- *Mislabeled padding bytes.* Many compilers generate padding bytes between consecutive blocks of code for alignment purposes. These bytes are not reached by typical runs, nor accessed as data, so their proper classification is ambiguous. IDA Pro typically classifies them as data, but this can complicate some code analyses by introducing many spurious code-data boundaries in the disassembly. In addition, these bytes can later become reachable if the binary undergoes hotpatching [10]. We therefore argue that these bytes are more properly classified as code.
- *Flows from code to data.* IDA Pro disassemblies frequently contain data bytes immediately preceded by non-branching or conditionally branching instructions. This is almost always an error; either the code is not actually reachable (and is therefore data misidentified as code) or the data is reachable (and is therefore code misidentified as data). The only exception to this that we have observed in practice is when a call instruction targets a non-returning procedure, such as an exception handler or the system's process-abort function. Such call instructions can be immediately followed by data.

To provide a rough estimate of the classification accuracy of IDA Pro, we wrote scripts in IDAPython [7] that detect obvious errors made by IDA Pro in its disassemblies. Table 1 gives a list of the executables we tested and counts of the errors we identified for IDA Pro 5.5. The main heuristic we used to identify errors is the existence of a control-flow from code to data. Certain other errors were identified via manual inspection. It is interesting to note that most programs compiled using the Gnu family of compilers have little to no errors in their IDA Pro disassemblies. This is probably because Gnu compilers tend to yield binaries in which code and data are less interleaved, and they perform fewer aggressive binary-level optimizations that can result in code that is difficult to disassemble.

In this paper, we present a disassembly algorithm that combines the heuristics manually applied by experts during reverse engineering and a language model

Table 1. Statistics of IDA Pro 5.5 disassembly errors

File Name	Instructions	Mistakes
Mfc42.dll	355906	1216
Mplayerc.exe	830407	474
RevelationClient.exe	66447	36
Vmware.exe	364421	183

that can capture both short-range and long-range correlations between byte sequences. Experimental results demonstrate that our algorithm can identify and successfully label a large number of code sequences that are missed by IDA Pro.

2 A Language Model for Disassembling x86 Executables

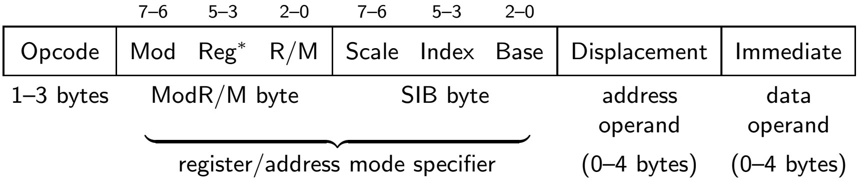
Without any debugging information at our disposal, we treat any given x86 executable as a string of arbitrary unsigned bytes. Our first task is to segment the single string into consecutive subsequences that are either code or data. A code-to-code, code-to-data, or data-to-code transition event occurs at each boundary between different instructions or between code and data.

The Intel architecture manual [11] specifies the decoding of each x86 instruction if the starting point for the instruction is known. Unfortunately, when code and data are interleaved it is not obvious whether a byte is the start of an instruction, the interior of an instruction, or a non-instruction (i.e., data). To tackle this problem we first decide whether a sequence of bytes is more likely to be code or data. The executable is then segmented using the opcodes defined in the Intel instruction encoding specification. Since we are unable to ensure a perfect segmentation, our next task is to classify each subsequence as code or data. Both tasks involve a context-based language model. We next formally describe each task and discuss the language model used in our disassembly algorithm.

2.1 Code Segmentation

In this section we first briefly review the x86 machine instruction set. We then define the code segmentation problem and present our algorithm to solve the problem.

Instruction Encodings. Figure 1 shows the x86 machine instruction binary format [11]. Instructions begin with 1–3 *opcode* bytes that identify the instruction. Instructions with operands are then followed by an *addressing form specifier* (ModR/M) byte that identifies register or memory operands for the instruction. Some addressing forms require a second *scale-index-base* (SIB) byte that specifies a memory addressing mode. The addressing mode essentially encodes a short formula that dynamically computes the memory operand at runtime. For example, addressing mode `[eax*4]+disp32` references a memory address obtained



*The Reg field is sometimes used as an opcode extension field.

Fig. 1. The x86 machine instruction format

by multiplying the contents of the `eax` register by 4 and then adding a 32-bit *displacement* constant. The displacement, if present, comes after the SIB byte. Finally, immediate operands (constants) are encoded last and have a width of up to 4 bytes (on 32-bit architectures).

In addition to this complicated instruction format, there are a number of prefix bytes that may precede the opcode bytes, all eleven of which may be used in combination. Some of these prefix bytes, if present, affect the length of the succeeding instruction's encoding by temporarily changing the default operand widths.

A few x86 machine instructions have multiple different correct representations at the assembly level. Most notable is the floating point `WAIT` instruction, which can either be interpreted as an opcode prefix for the instruction it precedes, or as a separate instruction in its own right. We adopt the former interpretation in our treatment, since it makes for a more compact assembly representation.

Problem Definition. We define the *tagging problem* as follows: Given a non-empty input string X over an alphabet Σ, find a set of transition events $\mathcal{T}^* = \{\$_1, \ldots, \$_M\}$ such that $\mathcal{T}^* = \arg\max_{\mathcal{T}} f(X, \mathcal{T})$, where $\$_i$ at position $i < |X|$ marks a transition event e in X, $\mathcal{T}$ denotes any possible set of transition events, and f is a function that measures the likelihood that X is tagged correctly.

The tagging problem resembles the word segmentation problem in some natural languages where no clear separations exist between different words [15]. In the word segmentation problem, the task is to find correct separations between sequences of characters to form words. In the tagging problem, our objective is to find separations between different instructions, and often between instructions and data as well. In both problems, resolving ambiguities is the major challenge. For example, a byte sequence `E8 F9 33 6A 00` can be a 5-byte call instruction (opcode `E8`), or three bytes of data followed by a push instruction (opcode `6A`). Ambiguities can only be resolved through investigating their surrounding context.

Solutions to the tagging problem must also successfully identify and ignore "noise" in the form of padding bytes. Padding bytes are neither executed as code nor accessed as data on any run of the executable, so their classification is ambiguous. However, reliably distinguishing these padding sequences from true code and data is highly non-trivial because the same sequence of bytes often appears as both code and padding within the same executable. For example, the instruction

```
8D A4 24 00 00 00 00    lea esp, [esp+0x0]
```

is semantically a *no-operation* (NOP), and is therefore used as padding within some instruction streams to align subsequent bytes to a cache line boundary, but is used in other instruction streams as a genuinely reachable instruction. Another common use of semantic NOPs is to introduce obfuscation to hide what the program is doing.

In general, code and data bytes may differ only in their locations in the sequence, not in their values. Any byte sequence that is code could appear as data in an executable, even though it should statistically appear much more often as code than data. Not every data sequence can be code, however, since not all byte sequences are legitimate instruction encodings.

The Tagging Algorithm. There are two components in our tagging algorithm: an instruction reference array and a utility function. The reference array stores the length of an instruction given the bytes of an opcode (and the existence of length-relevant prefix bytes). The utility function estimates the probability that a byte sequence is code. We estimate the probability using a context-based language model built from pre-tagged x86 executables.

Instruction Reference Array. From the x86 instruction decoding specification we derive a mapping from the bytes of an opcode to the length of the instruction. This is helpful in two respects: First, it marks a definite ending of an instruction that allows us to move directly to the next instruction or data. Second, it tells us when a series of bytes is undefined in the x86 instruction set, which means that the current byte cannot be the beginning of an instruction. We tested our code against more than ten million instructions in the IDA Pro disassembler and had 100% accurate instruction lengths.

Utility Function. The utility function helps predict whether a byte sequence is code or data in the current context. If the current byte sequence is unlikely to be code, our tagging algorithm moves to the next byte sequence. If we predict that the byte sequence is code, we look up the length of the instruction in the instruction reference array and move to the next byte sequence. The following two properties express the desired relationship between the utility function and its input byte sequence.

Property 1. A byte sequence bordered by transitions is tagged as code (resp., data) if its utility as code (resp., data) is greater than its utility as data (resp., code).

Property 2. A transition between two byte sequences S_A and S_B entails a semantic ordering in machine code: $f(S_B|S_A) \geq f(S_B|S_*)$, where S_* is any subsequence but S_A in a given binary, and f is the utility function.

Our utility function estimates the likelihood of a transition event using context-based analysis. We collect context statistics from a set of pre-tagged binaries in the training set. In a pre-tagged binary, code-code and code-data/data-code transitions are given. Two important forms of information are yielded by pre-tagged binaries. First, they provide semantic groupings of byte sequences that are either code or data; and second, they provide a semantic ordering between two subsequences, which predicts how likely a subsequence is followed by another. To correctly tag an input hex string, both pieces of information are important. This calls for a language model that

- can capture local coherence in a byte sequence, and
- can capture long-range correlations between two adjacent subsequences—i.e., subsequences separated by a code-code or code-data/data-code transition.

Several modern statistical data compression models [14] are known for their context-based analysis. These data models can work directly on any raw input regardless of source and type. We use the current state of the art data compression model as our language model. Before we discuss the details of the language model, we give the tagging algorithm in Algorithm 1.

Algorithm 1. Tagging

```
Input: $x_0 \ldots x_i \ldots x_{n-1}$                          // input string of bytes
       $M_c$                                                     // language model
Output: $x_0 \ldots x_i | x_{i+1} \ldots x_j | \cdots | x_k \ldots x_{n-1}$   // segmented string
$t \leftarrow 0$
while $t < n$ do
    $\ell \leftarrow 0$
    if $x_t \in M_c$ then
        $\ell \leftarrow codeLength(x_t \ldots x_{\min\{t+4,n-1\}})$   // lookup instruction length
    if $(\ell = 0) \vee (t + \ell > n)$ then $\ell \leftarrow 1$        // tag as possible data
    print $x_t \ldots x_{t+\ell-1}$                                // output the segment
    $t \leftarrow t + \ell$
```

2.2 Context-Based Data Compression Model

The compression model we use to store context statistics is *predication by partial matching* (PPM) [4,5,3]. The theoretical foundation of the PPM algorithm is the kth order Markov model, where k constrains the maximum order context based on which a symbol probability is predicted. PPM models both short-range and long-range correlations among subsequences by using dynamic context match. The context of the ith symbol x_i in an input string is the previous $i-1$ symbols. Its kth order context c_i^k includes only the k prior symbols. To predict the probability of seeing x_i in the current location of the input, the PPM algorithm first searches for a match of c_i^k in the context tree. If a match is found, $p(x_i|c_i^k)$ is returned as the symbol probability. If such a match does not exist in the context tree, an *escape event* is recorded and the model falls back to a lower-order context c_i^{k-1}. If a match is found, the following symbol probability is returned:

$$p(x_i|c_i^k) = p(Esc|c_i^k) \cdot p(x_i|c_i^{k-1})$$

where $p(Esc|c_i^k)$ is the escape probability conditioned on context c_i^k. The *escape probability* models the probability that x_i will be found in the lower-order context. This process is repeated whenever a match is not found until an order-0 context has been reached. If x_i appears in the input string for the first time, a uniform probability of distinct symbols that have been observed so far will be returned. Therefore, the probability of x_i in a string of input is modeled as follows:

$$p(x_i|c_i^k) = \begin{cases} \left(\prod_{j=k'+1}^{k} p(Esc|c_i^j)\right) \cdot p(x_i|c_i^{k'}) & \text{if } k \geq 0 \\ \frac{1}{|A|} & \text{if } k = -1 \end{cases}$$

where $k' \leq k$ is the context order when the first match is found for x_i, and $|A|$ is the number of distinct symbols seen so far in the input. If the symbol is not predicted by the order-0 model, a probability defined for the order -1 context is predicted.

The PPM model predicts symbol probabilities. To estimate the probability of a sequence of symbols, we compute the product of the symbol probabilities in the sequence. Thus, given a data sequence $X = x_1 x_2 \ldots x_d$ of length d, where x_i is a symbol in the alphabet, the probability of seeing the entire sequence given a compression model M can be estimated as

$$p(X|M) = \prod_{i=1}^{d} p(x_i|x_{i-k}^{i-1})$$

where $x_i^j = x_i x_{i+1} x_{i+2} \ldots x_j$ for $i < j$.

We use the above probability estimate as our utility function. We build two compression models M_c and M_d from the pre-tagged binaries in the training set: M_c is built from tagged instructions and M_d is built from tagged data. Given a new binary executable e and a subsequence e_i in e,

$$M_c = \{e_i \,|\, p(e_i|M_c) > p(e_i|M_d)\}$$

2.3 Classification

After tagging the transitions in the executable, we have segments of bytes. Even though the tagging algorithm outputs each segment either as code or data, we cannot assume this preliminary classification is correct because some data bytes may match legitimate opcodes for which a valid instruction length exists in the reference array. The tagging algorithm will output this segment as code even though it is data. Therefore, we need to reclassify each segment as data or code.

Our classification algorithm makes use of the aforementioned language model and several well known semantic heuristics. The language models are also used in the tagging algorithm. The heuristics are adapted from those used by human experts for debugging disassembly errors. We first discuss the language model-based classification module followed by the semantic heuristics.

Classification Using Language Model. Classifying byte sequences is a binary classification problem. We reuse the two compression models built for tagging. Recall that model M_c is built from pre-tagged code and model M_d is built from the pre-tagged data in the training set. To classify a byte sequence B, we compute a log likelihood of B using each data model $\alpha \in \{c, d\}$:

$$p(B|M_\alpha) = -\log \prod_{i=1}^{|B|} p(b_i | b_{i-k}^{i-1}, M_\alpha)$$

where M_α is the compression model associated with class α, $|B|$ is the length of byte sequence B, sequence $b_{i-k}, \ldots, b_i$ is a subsequence in B, and k is the length of the context. The class membership α of B is predicted by minimizing the cross entropy [16,2]:

$$\alpha = \underset{\alpha \in \{c,d\}}{\arg\min} - \frac{1}{|B|} p(B|M_\alpha)$$

Classification Using Heuristics. In addition to our context-based language models, certain semantic heuristics are helpful in determining an accurate class membership of an x86 byte sequence. Reverse engineers rely heavily upon such heuristics when manually correcting flawed disassemblies.

Word data tables. Many static data blocks in code sections store tables of 4-byte integers. Often the majority of 4-byte integers in these tables have similar values, such as when the table is a method dispatch or jump table consisting of code addresses that mostly lie within a limited virtual address range. One way to quickly identify such tables is to examine the distribution of byte values at addresses that are 1 less than a multiple of 4. When these high-order bytes have low variance, the section is likely to be a data table rather than code, and is classified accordingly.

16-bit addressing modes. When classifying a byte sequence as code yields a disassembly densely populated by instructions with 16-bit operands (and the binary is a 32-bit executable), this indicates that the sequence may actually be data misclassified as code. Modern x86 architectures support the full 16-bit instruction set of earlier processor generations for backward compatability reasons, but these legacy instructions appear only occasionally in most modern 32-bit applications. The 16-bit instructions often have short binary encodings, causing them to appear with higher frequency in randomly generated byte sequences than they do in actual code.

Data after unconditional jumps. Control-flows from code to data are almost always disassembly errors; either the data is reachable and is therefore code, or the code is actually unreachable and is therefore data. Thus, data inside of a code section can only occur at the very beginning of the section or after a branch instruction—usually an unconditional jump or return instruction. It can occasionally also appear after a call instruction if the call never returns (e.g., the call targets an exception handler or process-abort function). This observation gives rise to the following heuristics:

- If an instruction is a non-jump, non-return surrounded by data, it is reclassified as data.
- If a byte sequence classified as data encodes an instruction known to be a semantic NOP, it is reclassified as code.

3 Experimental Results

We tested our disassembly algorithm on the 11 real-world programs listed in Table 2. In each experiment, we used 10 of the programs to build the language models and the remaining one for testing. All the executables are pre-tagged using IDA Pro; however, IDA Pro yields imperfect disassemblies for all 11 executables. Some instructions it consistently labels as data, while others—particularly those that are semantic NOPs—it labels as data or code depending on the context. This leads to a noisy training set.

Table 2. Software programs for testing

File Name	File Size (K)	Code (K)	Data (K)	Transitions
`7zFM.exe`	379	271	3.3	1379
`notepad.exe`	68	23	8.6	182
`DosBox.exe`	3640	2947	67.2	15355
`WinRAR.exe`	1059	718	31.6	5171
`Mulberry.exe`	9276	4632	148.2	36435
`scummvm.exe`	11823	9798	49.2	47757
`emule.exe`	5624	3145	119.5	24297
`Mfc42.dll`	1110	751	265.5	15706
`Mplayerc.exe`	5858	4044	126.1	28760
`RevelationClient.exe`	382	252	18.4	1493
`Vmware.exe`	2675	1158	87.3	18259

Since we lack perfect disassemblies of any of these programs, evaluation of the classification accuracy of each algorithm is necessarily based on a manual comparison of the disassembly results. When the number of classification disagreements is large, this can quickly exceed the human processing limit. However, disagreements in which one algorithm identifies a large, contiguous code section missed by the other are relatively easy to verify by manual inspection. These constituted the majority of the disagreements, keeping the evaluation tractable.

3.1 Tagging Results

We first report the accuracy of our tagging algorithm. Inaccuracies can take the form of code misclassified as data (false negatives) and data misclassified as code (false positives). Both can have potentially severe consequences in the context of reverse engineering for malware defense. False negatives withhold potentially malicious code sequences from expert analysis, allowing attacks to succeed; false positives increase the volume of code that experts must examine, exacerbating the difficulty of separating potentially dangerous code from benign code. We therefore compute the tagging accuracy as

$$accuracy = 1 - \frac{false\ negatives + false\ positives}{total\ number\ of\ instructions}$$

where false positives count the number of instructions erroneously disassembled from data bytes.

As can be seen in Table 3 we were able to tag 6 of the 11 binaries with 100% accuracy. For the remaining 5, the tagging errors were mainly caused by misclassification of small word data tables (see §2.3) consisting of 12 or fewer bytes. Our heuristic for detecting such tables avoids matching such small tables in order to avoid misclassifying short semantic NOP sequences that frequently pad instruction sequences. Such padding often consists of 3 identical 4-byte instructions, which collectively resemble a very short word data table.

Table 3. Tagging accuracy

File Name	Errors	Total	Tagging Accuracy
`7zFM.exe`	0	88164	100%
`notepad.exe`	0	6984	100%
`DosBox.exe`	0	768768	100%
`WinRAR.exe`	39	215832	99.982%
`Mulberry.exe`	0	1437950	100%
`scummvm.exe`	0	2669967	100%
`emule.exe`	117	993159	99.988%
`Mfc42.dll`	0	355906	100%
`Mplayerc.exe`	307	830407	99.963%
`RevelationClient.exe`	71	66447	99.893%
`Vmware.exe`	16	364421	99.998%

3.2 Classification Results

To evaluate the classification accuracy we took the output of our tagging algorithm and ran each segment through the language model to get its class membership. Table 4 shows the classification results of our disassembly algorithm. False positives (FP), false negatives (FN), and overall classification accuracy is listed for each disassembler. False positives are subsequences that are data misclassified as code and false negatives are those that are code misclassified as data. As can be seen in Table 4 we were able to classify five of the 11 binaries with 100% accuracy.

3.3 eMule Case Study

To show some of the specific differences between decisions made by IDA Pro's disassembler and our approach, we here present a detailed case study of eMule,

Table 4. A comparison of mistakes made by IDA Pro and by our disassembler

	IDA Pro 5.5			Ours		
File Name	FP	FN	Accuracy	FP	FN	Accuracy
`7zFM.exe`	0	1	99.999%	0	0	100%
`notepad.exe`	4	0	99.943%	0	0	100%
`DosBox.exe`	0	26	99.997%	0	0	100%
`WinRAR.exe`	0	23	99.989%	0	39	99.982%
`Mulberry.exe`	0	202	99.986%	0	0	100%
`scummvm.exe`	0	65	99.998%	0	0	100%
`emule.exe`	0	681	99.931%	0	117	99.988%
`Mfc42.dll`	0	1216	99.658%	0	47	99.987%
`Mplayerc.exe`	0	2065	99.751%	0	307	99.963%
`RevelationClient.exe`	0	1781	97.320%	0	71	99.893%
`Vmware.exe`	0	183	99.950%	0	45	99.988%

a popular peer-to-peer file sharing program. Case studies for other executables in our test suite are similar to that presented here. Table 5 illustrates examples in which IDA Pro classified bytes were code but our disassembler determined that they were data, or vice versa. In the table, *db* is an assembly directive commonly used to mark data bytes in a code listing. To identify all discrepancies, we stored all instructions from both disassemblies to text files with code/data distinguishers before every instruction. We then used `sdiff` to find the differences. The cases in Table 5 summarize all of the different kinds of discrepancies we discovered.

IDA Pro makes heavy use of heuristic control-flow analysis to infer instruction start points in a sea of unclassified bytes. Thus, its classification of bytes immediately following a call instruction depends on its estimate of whether the called method could return. For example, Case 1 of Table 5 shows a non-returning call to an exception handler. The call is immediately followed by padding bytes that serve to align the body of the next function. These bytes are also legitimate (but unreachable) instructions, so could be classified as data or code (though we argue in §1 that a code classification is preferable). However, this control-flow analysis strategy leads to a classification error in Case 2 of the table, wherein IDA Pro incorrectly identifies method `GetDLGItem` as non-returning and therefore fails to disassemble the bytes that follow the call. Our disassembler correctly identifies both byte sequences as code. Such scenarios account for about 20 of IDA Pro's disassembly errors for eMule.

Case 3 of Table 5 illustrates a repetitive instruction sequence that is difficult to distinguish from a table of static data. IDA Pro therefore misidentifies some of the bytes in this sequence as data, whereas our algorithm correctly identifies all as code based on the surrounding context.

Many instruction sequences in x86 binaries are only reachable at runtime via dynamically computed jumps. These sequences are difficult to identify by control-flow analysis alone since the destinations of dynamic jumps cannot be statically predicted in general. Case 4 is an example where IDA Pro fails to identify a

Table 5. Disassembly discrepancies between IDA Pro and our disassembler for eMule

		Example Disassemblies	
Case	**Description**	**IDA Pro 5.5**	**Ours**

1	padding after a non-returning call	`call ExceptionHandler`
		db (1–9 bytes)
		`function_start`
			`call ExceptionHandler`
			code (1–9 bytes)
			`function_start`

2	calls misidentified as non-returning	`call GetDLGItem`
		db 88h 50h
		`sbb al, 8Bh`
			`call GetDLGItem`
			`mov edx, [eax+1Ch]`

3	repetitive instruction sequences	*db (4 bytes)*
		`push 0`
		`push 0`
		`call 429dd0h`
			`push 0`
			`push 0`
			`push 0`
			`push 0`
			`call 429dd0h`

4	missed computed jump targets	*db (12 bytes)*
		`push offset 41CC30h`
			`mov eax, large fs:0`
			`mov edx, [esp+8]`
			`push FFFFFFFFh`
			`push offset 41CC30h`

5	false computed jump targets	`push ecx`
		db FFh
		`adc eax, 7DFAB4h`
		`mov ebp, eax`
		db 8Bh
		`sbb esp, 0`
		db (13 bytes)
		`test esi, esi`
			`push ecx`
			`call 7DFAB4h`
			`mov ebp, eax`
			`mov eax, [ebx+0DCh]`
			`mov ecx, [eax+4]`
			`cmp ecx, esi`
			`jle loc_524D61`
			`test esi, esi`

6	missed opcode prefixes	`push offset 701268h`
		db 64h
		`mov eax, large ds:0`
			`push offset 701268h`
			`mov eax, large fs:0`

7	code following unconditional branches	`jmp 526396h`
		db 8Bh
		`or eax, 9CAF08h`
			`jmp 526396h`
			`mov ecx, 9CAF08h`

8	code following returns	`retn`
		db C4h 83h
		`sub al, CDh`
		`push es`
			`retn`
			`add esp, 2Ch`
			`int 6`

9	code following conditional branches	`jz 52518Fh`
		db 8Bh
		`or eax, 9CAF04h`
			`jz 52518F`
			`mov ecx, 9CAF04h`

computed jump target and therefore fails to classify the bytes at that address as code; however, our disassembler finds and correctly disassembles the instructions.

Misidentifying non-jump targets as possible targets leads to a different form of disassembly error. Case 5 illustrates an example in which an early phase of IDA Pro's analysis incorrectly identifies the interior byte of an instruction as a possible computed jump destination (probably because some bytes in a data section happened to encode that address). The bytes at that address disassemble to an `adc` instruction that turns out to be misaligned with respect to the surrounding sequence. This leads to an inconsistent mix of code and data that IDA Pro cannot reconcile because it cannot determine which interpretation of the bytes is correct. In contrast, our algorithm infers the correct instruction sequence, given in the rightmost column of the table.

Some instructions include prefix bytes, as discussed in §2.1. The suffix without the prefix bytes is itself a valid instruction encoding. IDA Pro's analysis sometimes misses these prefix bytes because it discovers the suffix encoding first and treats it as a self-contained instruction. This leads to the disassembly error depicted in Case 6 of the table. Our approach avoids this kind of error in all cases.

Cases 7–8 of the table illustrate disassembly errors in which IDA Pro fails to identify code bytes immediately following unconditional jumps and returns. These too are a consequence of relying too heavily on control-flow analysis to discover code bytes. Occasionally these errors even appear after conditional jumps, as shown in Case 9. It is unclear why IDA Pro makes this final kind of mistake, though we speculate that it may be the result of a dataflow analysis that incorrectly infers that certain conditional branches are always taken and therefore never fall through. Use of conditional branches as unconditional jumps is a common malware obfuscation technique that this analysis may be intended to counter. However, in this case it backfires and leads to an incorrect disassembly. Our method yields the correct disassembly on the right.

4 Conclusion

We developed and evaluated an automated disassembler using context-aware language models to separate instructions from instructions and code from data. Each segment in the resulting byte sequence is then separately classified as code or data. Evaluation of the technique demonstrates that our algorithm consistently yields more accurate disassemblies than the IDA Pro disassembler, which is widely regarded as the best commercial disassembly tool currently available.

Future work includes blending more sophisticated heuristics into our learning model, and trying block entropy approaches to better estimate the boundary between code and data.

In addition, larger-scale evaluation of our results could be facilitated by automating more of the evaluation process. One possible approach is to generate test binaries with perfect labels by using compiler options that artificially separate code from data, or that yield binary debugging information that can be used to infer correct labels. Unfortunately, most compilers and compiler modes that yield binaries for which the disassembly task is non-trivial are specifically those compilers that are not easy to modify (e.g., non-open source compilers) and those modes that do not support debugging (e.g., highly optimizing release modes). Pursuing this approach therefore requires identifying a suitable compiler.

We also plan to apply our disassembly technique to support more effective and reliable analysis and instrumentation of x86 binaries without source code for security purposes [8].

References

1. Balakrishnan, G., Gruian, R., Reps, T., Teitelbaum, T.: CodeSurfer/x86—a platform for analyzing x86 executables. In: Proceedings of the 14th International Conference on Compiler Construction (CC), pp. 250–254 (2005)
2. Bratko, A., Cormack, G.V., Filipič, B., Lynam, T.R., Zupan, B.: Spam filtering using statistical data compression models. Journal of Machine Learning Research 7, 2673–2698 (2006)
3. Cleary, J.G., Teahan, W.J.: Unbounded length contexts for PPM. The Computer Journal 40(2/3), 67–75 (1997)
4. Cleary, J.G., Witten, I.H.: Data compression using adaptive coding and partial string matching. IEEE Transactions on Communications 32(4), 396–402 (1984)
5. Cormack, G.V., Horspool, R.N.: Data compression using dynamic Markov modeling. The Computer Journal 30(6), 541–550 (1987)
6. Eagle, C.: The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler. No Starch Press, Inc., San Francisco (2008)
7. Erdélyi, G.: IDAPython: User scripting for a complex application. Bachelor's thesis, EVTEK University of Applied Sciences (2008)
8. Hamlen, K.W., Mohan, V., Wartell, R.: Reining in Windows API abuses with inlined reference monitors. Tech. Rep. UTDCS-18-10, The University of Texas at Dallas, Richardson, Texas (June 2010)
9. Hex-Rays: The IDA Pro disassembler and debugger (2011), www.hex-rays.com/idapro
10. Hunt, G., Brubacher, D.: Detours: Binary interception of Win32 functions. In: Proceedings of the 3rd USENIX Windows NT Symposium (WINSYM), pp. 14–21 (1999)
11. Intel: Intel® 64 and IA-32 Architectures Software Developer's Manual, vol. 2A & 2B: Instruction Set Reference. Intel Corporation (2011)
12. Kinder, J., Veith, H.: Jakstab: A static analysis platform for binaries. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 423–427. Springer, Heidelberg (2008)
13. Kinder, J., Zuleger, F., Veith, H.: An abstract interpretation-based framework for control flow reconstruction from binaries. In: Jones, N.D., Müller-Olm, M. (eds.) VMCAI 2009. LNCS, vol. 5403, pp. 214–228. Springer, Heidelberg (2009)
14. Moffat, A., Turpin, A.: Compression and Coding Algorithms. Kluwer Academic Publishers, Boston (2002)
15. Teahan, W.J., Wen, Y., McNab, R.J., Witten, I.H.: A compression-based algorithm for Chinese word segmentation. Computational Linguistics 26(3), 375–393 (2000)
16. Teahan, W.J.: Text classification and segmentation using minimum cross-entropy. In: Proceedings of the 6th International Conference on Computer-Assisted Information Retrieval (RIAO), pp. 943–961 (2000)

Bayesian Matrix Co-Factorization: Variational Algorithm and Cramér-Rao Bound

Jiho Yoo[1] and Seungjin Choi[1,2]

[1] Department of Computer Science
[2] Division of IT Convergence Engineering,
Pohang University of Science and Technology,
San 31 Hyoja-dong, Nam-gu, Pohang 790-784, Korea
{zentasis,seungjin}@postech.ac.kr

Abstract. Matrix factorization is a popular method for collaborative prediction, where unknown ratings are predicted by user and item factor matrices which are determined to approximate a user-item matrix as their product. Bayesian matrix factorization is preferred over other methods for collaborative filtering, since Bayesian approach alleviates overfitting, integrating out all model parameters using variational inference or sampling methods. However, Bayesian matrix factorization still suffers from the cold-start problem where predictions of ratings for new items or of new users' preferences are required. In this paper we present *Bayesian matrix co-factorization* as an approach to exploiting side information such as content information and demographic user data, where multiple data matrices are jointly decomposed, i.e., each Bayesian decomposition is coupled by sharing some factor matrices. We derive variational inference algorithm for Bayesian matrix co-factorization. In addition, we compute Bayesian Cramér-Rao bound in the case of Gaussian likelihood, showing that Bayesian matrix co-factorization indeed improves the reconstruction over Bayesian factorization of single data matrix. Numerical experiments demonstrate the useful behavior of Bayesian matrix co-factorization in the case of cold-start problems.

1 Introduction

Matrix factorization is a method for seeking a low-rank latent structure of data, approximating the data matrix as a product of two or more factor matrices. Matrix factorization is popular for collaborative prediction, where unknown ratings are predicted by user and item factor matrices which are determined to approximate a user-item matrix as their product [6,8,4,5,11,2]. Probabilistic matrix factorization was introduced in [8], in which a linear model with Gaussian observations was considered to learn user-specific and term-specific latent features, which became equivalent to the minimization of sum-of-squared errors with quadratic regularization terms. Bayesian approaches to matrix factorization are proposed based on the approximate inference such as the variational inference [4] or sampling [7], since the exact inference for the probabilistic model

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 537–552, 2011.

is intractable. Bayesian matrix factorization is preferred over other methods for collaborative filtering, since Bayesian approach alleviates overfitting by integrating out all model parameters.

Collaborative prediction algorithms suffer from the cold-start problem, where the users or items do not have sufficient number of given ratings. The cold-start problem commonly occurs in applying collaborative prediction in the practical problems because new users and new items, which has no previously given ratings, are continuously added to the system. Moreover, the users do not have high intention to rate the items remain in the system with small number of ratings of their own. The prediction accuracy of the collaborative prediction algorithm is seriously degraded because the algorithm only exploits the ratings given by the target users or items. To remedy the cold-start problem, efficient use of side information, such as item content information and user demographic information is crucial. Constrained probabilistic matrix factorization [8] is a representative method to incorporate side information into collaborative prediction based on matrix factorization, but it does not have clear relationship between the entity-relationship model of the whole data, so exploiting various kind of side information is not straight-forward.

Matrix co-factorization provides a way to systematically exploit the side information from the additional matrices. Matrix co-factorization jointly decomposes multiple data matrices, where each decomposition is coupled by sharing some factor matrices. Matrix co-factorization has been used to improve the performance of matrix factorization by incorporating knowledge in the additional matrices, such as label information [16], link information [17], and inter-subject variations [3]. One of the advantages of the matrix co-factorization is that it can be applied for the general entity-relationship models of the target data and the additional data [9,14], where the factor matrices correspond to the entities and the input matrices correspond to the relationships of the model. Since the entity-relationship model is a fundamental tool to model the relational data, this simple mapping between the entity-relationship model and the co-factorization model enables the straight-forward use of various kind of side information, especially for the cold-start problems where both the user side information and the item side information are required. Recently, Cramér-Rao bound (CRB) was computed for matix co-factorization with Gaussian likelihood on compressed sensing, showing that CRB is improved over matrix factorization, in the sense of reconstruction error when side information is incorporated into co-factorization [15].

We present a Bayesian matrix co-factorization (BMCF) to exploit side information, such as content information and user demographic data, into collaborative prediction problem to remedy the cold-start problems. We derive variational inference algorithm for BMCF. Sampling method is another possible approach for the BMCF [10], however the posterior computation requires storing multiple number of samples which is not appropriate for the large-scale collaborative prediction problems. A variational Bayesian approach for matrix co-factorization was mentioned in [13] without any details, so in this paper we provide the descriptions of the specific probabilistic model and the computation of variational posteriors, hyperparameters, and the predictive distributions.

In addition, we compute Bayesian Cramér-Rao bound (BCRB) for the BMCF model. BCRB provides a lower bound on the variance of any parametric estimators, even for the unbiased ones [12]. We compute the bound for the reconstruction error based on the BCRB, to show that BMCF indeed improves the reconstruction over Bayesian matrix factorization (BMF) of single data matrix. Numerical experiments confirm the improvements of the theoretical performance from BCRB, and demonstrate the useful behavior of BMCF in cold-start cases.

2 Bayesian Matrix Co-Factorization

The simplest case of matrix co-factorization deals with two input matrices, namely, the user-item rating matrix $\boldsymbol{X} \in \mathbb{R}^{I \times J}$ and the user-demographic information matrix $\boldsymbol{Y} \in \mathbb{R}^{I \times K}$. The input matrices are decomposed into the products of the following form,

$$\begin{aligned} \boldsymbol{X} &\approx \boldsymbol{U}^\top \boldsymbol{V}, \\ \boldsymbol{Y} &\approx \boldsymbol{U}^\top \boldsymbol{W}, \end{aligned}$$

where $\boldsymbol{U} \in \mathbb{R}^{D \times I}$ is the user factor matrix, $\boldsymbol{V} \in \mathbb{R}^{D \times J}$ is the item factor matrix, and $\boldsymbol{W} \in \mathbb{R}^{D \times K}$ is the demographic factor matrix. The user factor matrix $\boldsymbol{U}$ is shared in both decompositions, which makes it to be learned from the side information $\boldsymbol{Y}$ as well as the target ratings $\boldsymbol{X}$. The use of information in $\boldsymbol{Y}$ makes possible to predict meaningful ratings where $\boldsymbol{X}$ has extremely small number of given ratings.

To set up the probabilistic model for the co-factorizations, each element of the input matrices is modeled with the additive Gaussian noises, such as

$$\begin{aligned} x_{ij} &= \boldsymbol{u}_i^\top \boldsymbol{v}_j + \varepsilon_{ij}^{(x)}, \text{ for all } (i,j) \in \mathcal{O}^{(x)}, \\ y_{ik} &= \boldsymbol{u}_i^\top \boldsymbol{w}_k + \varepsilon_{ik}^{(y)}, \text{ for all } (i,j) \in \mathcal{O}^{(y)}, \end{aligned}$$

where $\boldsymbol{u}_i$ represents the i-th column of $\boldsymbol{U}$, $\boldsymbol{v}_j$ represents the j-th column of $\boldsymbol{V}$, and $\boldsymbol{w}_k$ represents the k-th column of $\boldsymbol{W}$. $\mathcal{O}^{(x)}$ and $\mathcal{O}^{(y)}$ denote the set of all indices of observed elements in $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. The additive noise $\varepsilon_{ij}^{(x)}$ and $\varepsilon_{ik}^{(y)}$ are modeled with the Gaussian distribution, such as

$$\begin{aligned} \varepsilon_{ij}^{(x)} &\sim \mathcal{N}(\varepsilon_{ij}^{(x)} | 0, \rho^{(x)}), \\ \varepsilon_{ik}^{(y)} &\sim \mathcal{N}(\varepsilon_{ik}^{(y)} | 0, \rho^{(y)}), \end{aligned}$$

where $\mathcal{N}(x|\mu, \rho)$ represents the Gaussian distribution with mean μ and the variance ρ, and $\rho^{(x)}$ and $\rho^{(y)}$ represent the noise variances for $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. Then, the likelihood of the co-factorization is modeled as

$$\begin{aligned} p(\boldsymbol{X}, \boldsymbol{Y} | \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) &= p(\boldsymbol{X} | \boldsymbol{U}, \boldsymbol{V}) p(\boldsymbol{Y} | \boldsymbol{U}, \boldsymbol{W}) \\ &= \prod_{(i,j) \in \mathcal{O}^{(x)}} \mathcal{N}(x_{ij} | \boldsymbol{u}_i^\top \boldsymbol{v}_j, \rho^{(x)}) \prod_{(i,k) \in \mathcal{O}^{(y)}} \mathcal{N}(y_{ik} | \boldsymbol{u}_i^\top \boldsymbol{w}_k, \rho^{(y)}). \end{aligned}$$

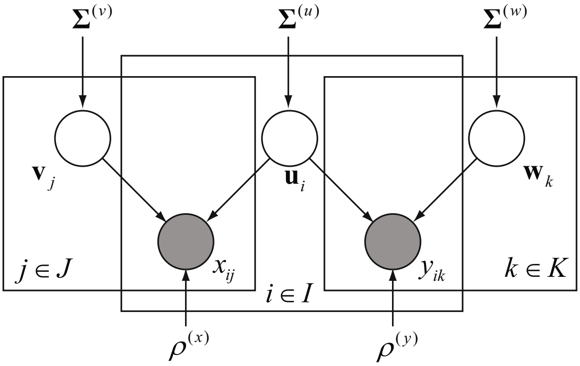


Fig. 1. The graphical model representation of the Bayesian matrix co-factorizations, where a side information matrix $\boldsymbol{Y}$ is available with the target matrix $\boldsymbol{X}$

The prior probabilities for the factor matrices are modeled with Gaussian,

$$\begin{aligned} p(\boldsymbol{U}) &= \prod_i \mathcal{N}(\boldsymbol{u}_i|\mathbf{0}, \boldsymbol{\Sigma}^{(u)}) = \prod_d \prod_i \mathcal{N}(u_{di}|0, \rho_d^{(u)}), \\ p(\boldsymbol{V}) &= \prod_j \mathcal{N}(\boldsymbol{v}_j|\mathbf{0}, \boldsymbol{\Sigma}^{(v)}) = \prod_d \prod_j \mathcal{N}(v_{dj}|0, \rho_d^{(v)}), \\ p(\boldsymbol{W}) &= \prod_k \mathcal{N}(\boldsymbol{w}_k|\mathbf{0}, \boldsymbol{\Sigma}^{(w)}) = \prod_d \prod_k \mathcal{N}(w_{dk}|0, \rho_d^{(w)}), \end{aligned}$$

where $\boldsymbol{\Sigma}^{(u)}$, $\boldsymbol{\Sigma}^{(v)}$ and $\boldsymbol{\Sigma}^{(w)}$ are the diagonal covariance matrices with the d-th diagonal element $\rho_d^{(u)}$, $\rho_d^{(v)}$, and $\rho_d^{(w)}$, respectively. Fig. 1 shows the graphical model representation of the probabilistic model.

We use the variational Bayesian approach to compute the posterior probability of each factor matrix. The lower-bound of the log of the marginal likelihood is computed by the Jensen's inequality with the functional $\mathcal{F}(q)$ of the auxiliary function $q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})$, such as

$$\begin{aligned} \log p(\boldsymbol{X}, \boldsymbol{Y}) &= \log \int\int\int q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) \frac{p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})}{q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})} d\boldsymbol{U} d\boldsymbol{V} d\boldsymbol{W} \\ &\geq \int\int\int q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) \log \frac{p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})}{q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})} d\boldsymbol{U} d\boldsymbol{V} d\boldsymbol{W} \\ &\equiv \mathcal{F}(q). \end{aligned}$$

In the variational Bayesian framework, we assume that the auxiliary function is further factorized into

$$q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) = q_u(\boldsymbol{U}) q_v(\boldsymbol{V}) q_w(\boldsymbol{W}),$$

leading to

$$\mathcal{F}(q_u, q_v, q_w) = \int\int\int q_u(\boldsymbol{U}) q_v(\boldsymbol{V}) q_w(\boldsymbol{W}) \log \frac{p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})}{q_u(\boldsymbol{U}) q_v(\boldsymbol{V}) q_w(\boldsymbol{W})} d\boldsymbol{U} d\boldsymbol{V} d\boldsymbol{W},$$

and $-\mathcal{F}(q_u, q_v, q_w)$ is referred to as *variational free energy.*

2.1 Updating Factor Matrices

In the variational Bayesian framework, the variational posteriors of the factor matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are computed with the following iterative updates,

$$q_u(\boldsymbol{U}) = \frac{1}{Z_u} \exp\left[\mathbb{E}_{V,W}\left\{\log p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})\right\}\right], \tag{1}$$

$$q_v(\boldsymbol{V}) = \frac{1}{Z_v} \exp\left[\mathbb{E}_{U,W}\left\{\log p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})\right\}\right], \tag{2}$$

$$q_w(\boldsymbol{W}) = \frac{1}{Z_w} \exp\left[\mathbb{E}_{U,V}\left\{\log p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})\right\}\right]. \tag{3}$$

To compute the variational posterior $q_u(\boldsymbol{U})$, the expectation over $\boldsymbol{V}$ and $\boldsymbol{W}$ is computed for the terms related to $\boldsymbol{U}$, which is written by

$$\begin{aligned}
&\mathbb{E}_{V,W}\left\{\log p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})\right\} \\
&= -\frac{1}{2}\sum_i \left[\boldsymbol{u}_i^\top \left(\left(\boldsymbol{\Sigma}^{(u)}\right)^{-1} + \frac{1}{\rho^{(x)}} \sum_{\substack{j|(i,j) \\ \in\mathcal{O}^{(x)}}} \left\langle \boldsymbol{v}_j \boldsymbol{v}_j^\top \right\rangle + \frac{1}{\rho^{(y)}} \sum_{\substack{k|(i,k) \\ \in\mathcal{O}^{(y)}}} \left\langle \boldsymbol{w}_k \boldsymbol{w}_k^\top \right\rangle \right) \boldsymbol{u}_i \right] \\
&\quad -\frac{1}{2}\sum_i \left[-2 \left(\frac{1}{\rho^{(x)}} \sum_{\substack{j|(i,j) \\ \in\mathcal{O}^{(x)}}} x_{ij} \left\langle \boldsymbol{v}_j \right\rangle^\top + \frac{1}{\rho^{(y)}} \sum_{\substack{k|(i,k) \\ \in\mathcal{O}^{(y)}}} y_{ik} \left\langle \boldsymbol{w}_k \right\rangle^\top \right) \boldsymbol{u}_i \right] + C,
\end{aligned}$$

where $\langle\cdot\rangle$ represents the expectation. From (1), the derivation leads to the variational posterior of $\boldsymbol{U}$ in the following form,

$$q_u(\boldsymbol{U}) \sim \prod_i \mathcal{N}\left(\boldsymbol{u}_i | \boldsymbol{\mu}_i^{(u)}, \boldsymbol{\Phi}_i^{(u)}\right),$$

where

$$\boldsymbol{\mu}_i^{(u)} = \boldsymbol{\Phi}_i^{(u)} \left(\frac{1}{\rho^{(x)}} \sum_{j|(i,j)\in\mathcal{O}^{(x)}} x_{ij} \left\langle \boldsymbol{v}_j \right\rangle + \frac{1}{\rho^{(y)}} \sum_{k|(i,k)\in\mathcal{O}^{(y)}} y_{ik} \left\langle \boldsymbol{w}_k \right\rangle \right),$$

$$\left(\boldsymbol{\Phi}_i^{(u)}\right)^{-1} = \left(\boldsymbol{\Sigma}^{(u)}\right)^{-1} + \frac{1}{\rho^{(x)}} \sum_{j|(i,j)\in\mathcal{O}^{(x)}} \left\langle \boldsymbol{v}_j \boldsymbol{v}_j^\top \right\rangle + \frac{1}{\rho^{(y)}} \sum_{k|(i,k)\in\mathcal{O}^{(y)}} \left\langle \boldsymbol{w}_k \boldsymbol{w}_k^\top \right\rangle.$$

As stated before, the user factor matrix is updated by using the side information matrix $\boldsymbol{Y}$, as well as the rating matrix $\boldsymbol{X}$, which enables the learning in the cold-start situation where $\boldsymbol{X}$ has no given ratings for some users.

The variational posteriors for the factor matrices $\boldsymbol{V}$ is computed from (2), which becomes

$$q_v(\boldsymbol{V}) = \prod_j \mathcal{N}(\boldsymbol{v}_j | \boldsymbol{\mu}_j^{(v)}, \boldsymbol{\Phi}_j^{(v)}),$$

where

$$\boldsymbol{\mu}_j^{(v)} = \boldsymbol{\Phi}_j^{(v)} \left(\frac{1}{\rho^{(x)}} \sum_{i|(i,j)\in\mathcal{O}^{(x)}} x_{ij} \langle \boldsymbol{u}_i \rangle \right),$$

$$\left(\boldsymbol{\Phi}_j^{(v)}\right)^{-1} = \left(\boldsymbol{\Sigma}^{(v)}\right)^{-1} + \frac{1}{\rho^{(x)}} \sum_{i|(i,j)\in\mathcal{O}^{(x)}} \left\langle \boldsymbol{u}_i \boldsymbol{u}_i^\top \right\rangle .$$

Similarly from (3), the variational posterior of $\boldsymbol{W}$ is computed by

$$q_w(\boldsymbol{W}) = \prod_k \mathcal{N}(\boldsymbol{w}_k | \boldsymbol{\mu}_k^{(w)}, \boldsymbol{\Phi}_k^{(w)}),$$

where

$$\boldsymbol{\mu}_k^{(w)} = \boldsymbol{\Phi}_k^{(w)} \left(\frac{1}{\rho^{(y)}} \sum_{i|(i,k)\in\mathcal{O}^{(y)}} y_{ik} \langle \boldsymbol{u}_i \rangle \right),$$

$$\left(\boldsymbol{\Phi}_k^{(w)}\right)^{-1} = \left(\boldsymbol{\Sigma}^{(w)}\right)^{-1} + \frac{1}{\rho^{(y)}} \sum_{i|(i,k)\in\mathcal{O}^{(y)}} \left\langle \boldsymbol{u}_i \boldsymbol{u}_i^\top \right\rangle .$$

The sufficient statistics for the above posteriors are easily computed by using the properties of the Gaussian distribution. The sufficient statistics for $\boldsymbol{u}_i$ are computed as

$$\langle \boldsymbol{u}_i \rangle = \overline{\boldsymbol{\mu}}_i^{(u)},$$

$$\left\langle \boldsymbol{u}_i \boldsymbol{u}_i^\top \right\rangle = \overline{\boldsymbol{\Sigma}}_i^{(u)} + \overline{\boldsymbol{\mu}}_i^{(u)} \overline{\boldsymbol{\mu}}_i^{(u)\top},$$

and the sufficient statistics for $\boldsymbol{v}_j$ and $\boldsymbol{w}_k$ are computed in the similar forms.

2.2 Learning Hyperparameters

We use the empirical Bayes estimation to update hyperparameters $\rho^{(x)}$, $\rho^{(y)}$, $\boldsymbol{\Sigma}^{(u)}$, $\boldsymbol{\Sigma}^{(v)}$ and $\boldsymbol{\Sigma}^{(w)}$. The variational free energy $\mathcal{F}(q_u, q_v, q_w)$ is used to compute the point estimate of the hyperparameters.

Taking derivative of the variational free energy with respect to $\rho^{(x)}$ leads

$$\frac{\partial \mathcal{F}(q_u, q_v, q_w)}{\partial \rho^{(x)}} = -\frac{N^{(x)}}{2} \frac{1}{\rho^{(x)}} + \frac{1}{2(\rho^{(x)})^2} \sum_{(i,j)\in\mathcal{O}^{(x)}} \left\langle (x_{ij} - \boldsymbol{u}_i^\top \boldsymbol{v}_j)^2 \right\rangle ,$$

where $N^{(x)}$ represents the total number of observed entries in the matrix $\boldsymbol{X}$. Then, $\rho^{(x)}$ is computed by

$$\rho^{(x)} = \frac{1}{N^{(x)}} \sum_{(i,j)\in\mathcal{O}^{(x)}} \left\{ x_{ij}^2 - 2x_{ij} \langle \boldsymbol{u}_i \rangle^\top \langle \boldsymbol{v}_j \rangle + \operatorname{tr}\left(\left\langle \boldsymbol{u}_i \boldsymbol{u}_i^\top \right\rangle \left\langle \boldsymbol{v}_j \boldsymbol{v}_j^\top \right\rangle \right) \right\},$$

where $\mathrm{tr}(\cdot)$ represents the trace of the matrix. The update for $\rho^{(y)}$ is computed in the same way.

Taking derivative of $\mathcal{F}(q_u, q_v, q_w)$ with respect to $\rho_d^{(u)}$, which is the d-th diagonal element of $\boldsymbol{\Sigma}^{(u)}$, leads

$$\frac{\partial \mathcal{F}(q_u, q_v, q_w)}{\partial \rho_d^{(u)}} = -\frac{I}{2}\frac{1}{\rho_d^{(u)}} + \frac{1}{2\left(\rho_d^{(u)}\right)^2}\left[\sum_i \left\langle \boldsymbol{u}_i \boldsymbol{u}_i^\top \right\rangle\right]_{dd},$$

and set this to be zero leads the update

$$\rho_d^{(u)} = \frac{1}{I}\left[\sum_i \left\langle \boldsymbol{u}_i \boldsymbol{u}_i^\top \right\rangle\right]_{dd}.$$

The above update is re-written for $\boldsymbol{\Sigma}^{(u)}$ in the following form,

$$\boldsymbol{\Sigma}^{(u)} = \frac{1}{I}\mathrm{ddiag}\left(\sum_i \left\langle \boldsymbol{u}_i \boldsymbol{u}_i^\top \right\rangle\right),$$

where $\mathrm{ddiag}(\boldsymbol{A})$ represents the diagonal matrix consisting of the diagonal elements of the matrix $\boldsymbol{A}$. The update for $\boldsymbol{\Sigma}^{(v)}$ and $\boldsymbol{\Sigma}^{(w)}$ are derived in the similar way.

2.3 Predictive Distribution

There are two kinds of prediction tasks in the collaborative prediction problem: the *hold-out* prediction and the *fold-in* prediction. In the hold-out prediction, we want to predict a missing entry $x_{i^*j^*}$ in the input rating matrix $\boldsymbol{X}$, where $(i^*, j^*) \notin \mathcal{O}^{(x)}$. Then, the predictive distribution is calculated as

$$\begin{aligned} p(x_{i^*j^*}|\boldsymbol{X}) &= \int p(x_{i^*j^*}|\boldsymbol{U}, \boldsymbol{V}) q_u^*(\boldsymbol{U}) q_v^*(\boldsymbol{V}) d\boldsymbol{U} d\boldsymbol{V} \\ &= \mathcal{N}(x_{i^*j^*}|\left\langle \boldsymbol{u}_{i^*}\right\rangle^\top \left\langle \boldsymbol{v}_{j^*}\right\rangle, \rho^{(x)}). \end{aligned}$$

Therefore, the prediction becomes the product of corresponding columns of factor matrices, which is $\hat{x}_{i^*j^*} = \left\langle \boldsymbol{u}_{i^*}\right\rangle^\top \left\langle \boldsymbol{v}_{j^*}\right\rangle$.

In the fold-in prediction, we want to predict the rating value of new users or items. If we want to predict the rating $x_{i^+j^*}$ for the new user i^+, the predictive distribution is computed by

$$\begin{aligned} p(x_{i^+j^*}|\boldsymbol{X}, \boldsymbol{x}_{i^+}, \boldsymbol{Y}, \boldsymbol{y}_{i^+}) = \int\int\int & p(x_{i^+j^*}|\boldsymbol{u}_{i^+}, \boldsymbol{v}_{j^*}) p(\boldsymbol{u}_{i^+}|\boldsymbol{V}, \boldsymbol{x}_{i^+}, \boldsymbol{W}, \boldsymbol{y}_{i^+}) \\ & p(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}|\boldsymbol{X}, \boldsymbol{Y}) d\boldsymbol{U} d\boldsymbol{V} d\boldsymbol{W} d\boldsymbol{u}_{i^+}. \end{aligned}$$

The predictive distribution depends on the posterior distribution of the new factor, which is computed by using the Bayes' rule,

$$\begin{aligned} & \log p(\boldsymbol{u}_{i^+}|\boldsymbol{V}, \boldsymbol{x}_{i^+}, \boldsymbol{W}, \boldsymbol{y}_{i^+}) \\ &= \log p(\boldsymbol{x}_{i^+}|\boldsymbol{V}, \boldsymbol{u}_{i^+}) + \log p(\boldsymbol{y}_{i^+}|\boldsymbol{W}, \boldsymbol{u}_{i^+}) + \log p(\boldsymbol{u}_{i^+}) + C \\ &= \log \mathcal{N}(\boldsymbol{u}_{i^+}|\boldsymbol{\mu}_{i^+}^{(u)}, \boldsymbol{\Phi}_{i^+}^{(u)}), \end{aligned}$$

where

$$\left(\boldsymbol{\Phi}_{i+}^{(u)}\right)^{-1} = \left(\boldsymbol{\Sigma}^{(u)}\right)^{-1} + \frac{1}{\rho^{(x)}} \sum_{j|(i^+,j)\in\mathcal{O}^{(x)}} \left\langle \boldsymbol{v}_j \boldsymbol{v}_j^\top \right\rangle + \frac{1}{\rho^{(y)}} \sum_{k|(i^+,k)\in\mathcal{O}^{(y)}} \left\langle \boldsymbol{w}_k \boldsymbol{w}_k^\top \right\rangle,$$

$$\boldsymbol{\mu}_{i+}^{(u)} = \boldsymbol{\Phi}_{i+}^{(u)} \left(\frac{1}{\rho^{(x)}} \sum_{j|(i^+,j)\in\mathcal{O}^{(x)}} x_{i+j} \left\langle \boldsymbol{v}_j \right\rangle + \frac{1}{\rho^{(y)}} \sum_{k|(i^+,k)\in\mathcal{O}^{(y)}} y_{i+k} \left\langle \boldsymbol{w}_k \right\rangle \right).$$

This posterior indicates that the prediction is computed based on the observed ratings in $\boldsymbol{x}_{i+}$ and the additional information $\boldsymbol{y}_{i+}$, which makes the prediction in the cold-start situation possible. The unknown ranking in the fold-in case is predicted with the posterior distribution by $x_{i+j^*} = \left\langle \boldsymbol{u}_{i+} \right\rangle^\top \left\langle \boldsymbol{v}_{j^*} \right\rangle$, where $\left\langle \boldsymbol{u}_{i+} \right\rangle = \boldsymbol{\mu}_{i+}^{(u)}$.

2.4 BMCF for General Cases

So far we considered the simplest example of the co-factorization, which has three entities: user, item, and user demographic information, and two relationships: user-item ratings and user-demographic information. We generalize the results for the arbitrary entity-relationship model by mapping the entities to the factor matrices and relationships to the input matrices. In this way, co-factorization model is directly induced from the entity-relationship model of data, which enables straight-forward use of various kinds of side-information.

The entity-relationship model consists of entities, attributes for the entities, and relationships between the entities. For the one-to-one correspondence between the entity-relationship model and the co-factorization model, we eliminate the use of attributes by modeling them as a separate entity having relationship with the corresponding entity. Then, we use the entity-relationship model consists of the set of entities $\mathcal{E}$ and the set of relationships $\mathcal{R}$. The co-factorization model is built with the input matrices $\boldsymbol{X}^{(a,b)}$ for all relationships $(a,b) \in \mathcal{R}$ and

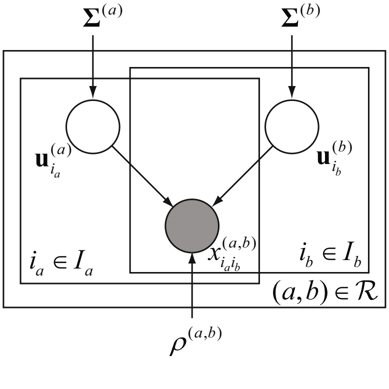


Fig. 2. The graphical model representation of the Bayesian matrix co-factorizations in the general case

Table 1. Model and the algorithms for the BMCF in general cases. We denote set of all input matrices as $\mathcal{X}$ and set of all factor matrices as $\mathcal{U}$. $I^{(a)}$ represents the number of columns in the factor matrix $\boldsymbol{U}^{(a)}$, and $N^{(a,b)}$ represents the number of observed entries in the input matrix $\boldsymbol{X}^{(a,b)}$.

Likelihood	$p(\mathcal{X}\mid\mathcal{U}) = \prod_{(a,b)\in\mathcal{R}} \prod_{(i_a,i_b)\in\mathcal{O}^{(a,b)}} \mathcal{N}(x^{(a,b)}_{i_a i_b} \mid \boldsymbol{u}^{(a)\top}_{i_a}\boldsymbol{u}^{(b)}_{i_b}, \rho^{(a,b)})$
Prior	$p(\boldsymbol{U}^{(a)}) = \prod_{i_a}\mathcal{N}(\boldsymbol{u}^{(a)}_{i_a}\mid\boldsymbol{0},\boldsymbol{\Sigma}^{(a)}) = \prod_d\prod_i \mathcal{N}(u_{di_a}\mid 0,\rho^{(a)}_d)$
Posterior	$q_a(\boldsymbol{U}^{(a)}) \sim \prod_{i_a}\mathcal{N}\left(\boldsymbol{u}^{(a)}_{i_a}\mid\boldsymbol{\mu}^{(a)}_{i_a},\boldsymbol{\Phi}^{(a)}_{i_a}\right)$, where $\boldsymbol{\mu}^{(a)}_{i_a} = \boldsymbol{\Phi}^{(a)}_{i_a}\left(\sum_{b\mid(a,b)\in\mathcal{R}}\sum_{i_b\mid(i_a,i_b)\in\mathcal{O}^{(a,b)}}\frac{1}{\rho^{(a,b)}}x^{(a,b)}_{i_a i_b}\left\langle\boldsymbol{u}^{(b)}_{i_b}\right\rangle\right)$ $\left(\boldsymbol{\Phi}^{(a)}_{i_a}\right)^{-1} = \left(\boldsymbol{\Sigma}^{(a)}\right)^{-1} + \sum_{b\mid(a,b)\in\mathcal{R}}\sum_{i_b\mid(i_a,i_b)\in\mathcal{O}^{(a,b)}}\frac{1}{\rho^{(a,b)}}\left\langle\boldsymbol{u}^{(b)}_{i_b}\boldsymbol{u}^{(b)\top}_{i_b}\right\rangle$
Sufficient statistics	$\left\langle\boldsymbol{u}^{(a)}_{i_a}\right\rangle = \boldsymbol{\mu}^{(a)}_{i_a}$ $\left\langle\boldsymbol{u}^{(a)}_{i_a}\boldsymbol{u}^{(a)\top}_{i_a}\right\rangle = \boldsymbol{\Phi}^{(a)}_{i_a} + \boldsymbol{\mu}^{(a)}_{i_a}\boldsymbol{\mu}^{(a)\top}_{i_a}$
Parameters	$\rho^{(a,b)} = \frac{1}{N^{(a,b)}}\sum_{(i_a,i_b)\in\mathcal{O}^{(a,b)}}\left\{\left(x^{(a,b)}_{i_a i_b}\right)^2 - 2x^{(a,b)}_{i_a i_b}\left\langle\boldsymbol{u}^{(a)}_{i_a}\right\rangle^\top\left\langle\boldsymbol{u}^{(b)}_{i_b}\right\rangle\right\}$ $+\frac{1}{N^{(a,b)}}\sum_{(i_a,i_b)\in\mathcal{O}^{(a,b)}}\left\{\mathrm{tr}\left(\left\langle\boldsymbol{u}^{(a)}_{i_a}\boldsymbol{u}^{(a)\top}_{i_a}\right\rangle\left\langle\boldsymbol{u}^{(b)}_{i_b}\boldsymbol{u}^{(b)\top}_{i_b}\right\rangle\right)\right\}$ $\boldsymbol{\Sigma}^{(a)} = \frac{1}{I^{(a)}}\mathrm{ddiag}\left(\sum_{i_a}\left\langle\boldsymbol{u}^{(a)}_{i_a}\boldsymbol{u}^{(a)\top}_{i_a}\right\rangle\right)$
Prediction	$x_{i^*_a i^*_b} = \left\langle\boldsymbol{u}^{(a)}_{i^*_a}\right\rangle^\top\left\langle\boldsymbol{u}^{(b)}_{i^*_b}\right\rangle$ In the fold-in case, using $\left\langle\boldsymbol{u}^{(a)}_{i^*_a}\right\rangle = \boldsymbol{\Phi}^{(a)-1}_{i^*_a}\left(\sum_{c\mid(a,c)\in\mathcal{R}}\left(\frac{1}{\rho^{(a,c)}}\sum_{i_c\mid(i^*_a,i_c)\in\mathcal{O}^{(a,c)}}x_{i^*_a i_c}\left\langle\boldsymbol{u}_{i_c}\right\rangle\right)\right)$ $\left(\boldsymbol{\Phi}^{(a)}_{i^*_a}\right)^{-1} = \left(\boldsymbol{\Sigma}^{(a)}\right)^{-1} + \sum_{c\mid(a,c)\in\mathcal{R}}\left(\frac{1}{\rho^{(a,c)}}\sum_{i_c\mid(i^*_a,i_c)\in\mathcal{O}^{(a,c)}}\left\langle\boldsymbol{u}^{(c)}_{i_c}\boldsymbol{u}^{(c)\top}_{i_c}\right\rangle\right)$

and the factor matrix $\boldsymbol{U}^{(a)}$ for all entities $a \in \mathcal{E}$. If we use the indices for a-th entity as i_a, the matrix co-factorization model is written by

$$x^{(a,b)}_{i_a i_b} = \boldsymbol{u}^{(a)\top}_{i_a}\boldsymbol{u}^{(b)}_{i_b} + \varepsilon^{(a,b)}_{i_a i_b} \text{ for all } (a,b)\in\mathcal{R}, (i_a,i_b)\in\mathcal{O}^{(a,b)},$$

where $\mathcal{O}^{(a,b)}$ represents the set of all observed entries in $\boldsymbol{X}^{(a,b)}$, and we used the additive Gaussian noise $\varepsilon^{(a,b)}_{i_a i_b}$ having distribution

$$\varepsilon^{(a,b)}_{i_a i_b} \sim \mathcal{N}(\varepsilon^{(a,b)}_{i_a i_b}\mid 0, \rho^{(a,b)}),$$

where $\rho^{(a,b)}$ represents the noise variance, which leads the Gaussian likelihood. The prior for each factor matrix $\boldsymbol{U}^{(a)}$ is modeled as Gaussian with zero mean and the variance $\rho^{(a)}_d$. The graphical model representation of the general matrix co-factorization is shown in the Fig. 2. The probabilistic model, update of factor matrices and hyperparameters, and the predictive distributions are summarized in Table 1.

3 Bayesian Cramér-Rao Bounds for Bayesian Matrix Co-Factorization

The Cramér-Rao Bound (CRB) places a lower bound on the variance of unbiased estimator for the deterministic parameters [1], as the inverse of the Fisher information matrix $\mathcal{F}$, which is written by,

$$\mathbb{E}\left\{(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^\top\right\} \geq \mathcal{F}^{-1},$$

where $\boldsymbol{\theta}$ is the estimated parameter and $\hat{\boldsymbol{\theta}}$ is the true value for it. Each element of the Fisher information matrix is computed by

$$\mathcal{F}_{ij} = \mathbb{E}_{\boldsymbol{x}}\left\{-\frac{\partial^2 \log p(\boldsymbol{x}|\boldsymbol{\theta})}{\partial\theta_i\partial\theta_j}\right\}.$$

The computation of Fisher information matrix mainly depends on the likelihood of the model.

On the other hand, the Bayesian Cramer-Rao bound (BCRB) or Posterior Cramer-Rao Bound [12] uses a different form of the Fisher information matrix, which depends on the joint probability of the observation and the parameters,

$$\mathcal{F}_{ij} = \mathbb{E}_{\boldsymbol{x},\theta}\left\{-\frac{\partial^2 \log p(\boldsymbol{x},\boldsymbol{\theta})}{\partial\theta_i\partial\theta_j}\right\}. \tag{4}$$

In this case we use the prior probability, as well as the likelihood, to compute the Fisher information matrix, and the expectation is also taken over the parameters. The benefit of using BCRB over CRB is that the BCRB is known to provide a lower bound on the variance of any parametric estimators, even for the unbiased ones [12]. In this section we use the BCRB to show the improvement of theoretical bounds of the proposed co-factorization model over the standard matrix factorization model.

3.1 Computation of Fisher Information Matrix

To compute the BCRB for the matrix co-factorization model, we rearrange the factor matrices to be a parameter vector. For example, if we have two factor matrices $\boldsymbol{U}^{(a)}$ and $\boldsymbol{U}^{(b)}$, the parameter vector $\boldsymbol{\theta}$ becomes

$$\boldsymbol{\theta} = [\boldsymbol{u}_1^{(a)\top} \cdots \boldsymbol{u}_{I^{(a)}}^{(a)\top} \boldsymbol{u}_1^{(b)\top} \cdots \boldsymbol{u}_{I^{(b)}}^{(b)\top}]^\top,$$

where $I^{(a)}$ represents the number of columns in the factor matrix $\boldsymbol{U}^{(a)}$. Then, each element of the Fisher information matrix is computed as (4). The log joint probability of BMCF is computed as the sum of log-likelihood and log priors (Table 1).

Case I $\left(\frac{\partial^2 \log p(\mathcal{X},\mathcal{U})}{\partial \boldsymbol{U}^{(a)}\partial \boldsymbol{U}^{(b)}}\right)$**:** In this case, we take the first derivative of the log joint probability with respect to a parameter $u^{(a)}_{d^* i^*_a}$ in the factor matrix $\boldsymbol{U}^{(a)}$, which becomes,

$$\frac{\partial \log p(\mathcal{X},\mathcal{U})}{\partial u^{(a)}_{d^* i^*_a}}$$
$$= \sum_{\substack{c|(a,c)\in\mathcal{R} \\ i_c|(i^*_a,i_c)\in\mathcal{O}^{(a,c)}}} \left[\frac{1}{\rho^{(a,c)}} x^{(a,c)}_{i^*_a i_c} u^{(c)}_{d^* i_c} - \frac{1}{\rho^{(a,c)}} u^{(c)}_{d^* i_c}\left(\sum_d u^{(a)}_{d i^*_a} u^{(c)}_{d i_c}\right)\right] - \frac{1}{\rho^{(a)}_{d^*}} u^{(a)}_{d^* i^*_a},$$

where $\mathcal{X}$ is the set of all input matrices and $\mathcal{U}$ is the set of all factor matrices. If we take the second derivative with respect to the parameter from the different factor matrix $\boldsymbol{U}^{(b)}$, which is $u^{(b)}_{d^+ i^+_b}$, it is written as

$$\frac{\partial^2 \log p(\mathcal{X},\mathcal{U})}{\partial u^{(a)}_{d^* i^*_a}\partial u^{(b)}_{d^+ i^+_b}} = \sum_{(i^*_a,i^+_b)\in\mathcal{O}^{(a,b)}} \left[-\frac{1}{\rho^{(a,b)}} u^{(a)}_{d^* i^*_a} u^{(b)}_{d^+ i^+_b}\right],$$

for $d^+ \neq d^*$. If $d^+ = d^*$, the second derivative is written as

$$\frac{\partial^2 \log p(\mathcal{X},\mathcal{U})}{\partial u^{(a)}_{d^* i^*_a}\partial u^{(b)}_{d^* i^+_b}} = \sum_{\substack{(i^*_a,i^+_b) \\ \in\mathcal{O}^{(a,b)}}} \left[\frac{1}{\rho^{(a,b)}}\left(x^{(a,b)}_{i^*_a i^+_b} - \sum_d u^{(a)}_{d i^*_a} u^{(b)}_{d i^+_b}\right) - \frac{1}{\rho^{(a,b)}} u^{(a)}_{d^* i^*_a} u^{(b)}_{d^* i^+_b}\right].$$

The expectations of above second derivatives vanish, so the elements of Fisher information matrix corresponding to the part also become zero.

Case II $\left(\frac{\partial^2 \log p(\mathcal{X},\mathcal{U})}{\partial \boldsymbol{U}^{(a)}\partial \boldsymbol{U}^{(a)}}\right)$**:** The second derivative with respect to the element $u^{(a)}_{d^+ i^+_a}$ from the same factor matrix $\boldsymbol{U}^{(a)}$ vanishes if $i^+_a \neq i^*_a$. If $i^+_a = i^*_a$ and $d^+ \neq d^*$,

$$\frac{\partial^2 \log p(\mathcal{X},\mathcal{U})}{\partial u^{(a)}_{d^* i^*_a}\partial u^{(a)}_{d^+ i^*_a}} = -\sum_{c|(a,c)\in\mathcal{R}} \frac{1}{\rho^{(a,c)}} \sum_{i_c|(i^*_a,i_c)\in\mathcal{O}^{(a,c)}} u^{(c)}_{d^* i_c} u^{(c)}_{d^+ i_c},$$

but the expectation of it vanishes.

The only nonzero second-derivative value is arisen if we differentiate with the same element from the same matrix, that is, in the case of $i^+_a = i^*_a$ and $d^+ = d^*$, which becomes

$$\frac{\partial^2 \log p(\mathcal{X},\mathcal{U})}{\partial u^{(a)}_{d^* i^*_a}\partial u^{(a)}_{d^* i^*_a}} = -\sum_{c|(a,c)\in\mathcal{R}} \frac{1}{\rho^{(a,c)}} \sum_{i_c|(i^*_a,i_c)\in\mathcal{O}^{(a,c)}} \left(u^{(c)}_{d^* i_c}\right)^2 - \frac{1}{\rho^{(a)}_{d^*}}.$$

The Fisher information matrix is computed as

$$\mathbb{E}_{X,U}\left\{-\frac{\partial^2 \log p(\mathcal{X},\mathcal{U})}{\partial u^{(a)}_{d^* i^*_a}\partial u^{(a)}_{d^* i^*_a}}\right\} = \sum_{c|(a,c)\in\mathcal{R}} \frac{N^{(a,c)}_{i^*_a}\rho^{(c)}_{d^*}}{\rho^{(a,c)}} + \frac{1}{\rho^{(a)}_{d^*}}, \qquad (5)$$

where $N_{i_a^*}^{(a,c)}$ represents the number of observed entries in the i_a^*-th column of the matrix $\boldsymbol{X}^{(a,c)}$. Because the only nonzero values come from the differentiating with the same parameter, the Fisher information matrix becomes a diagonal matrix.

If we use the standard matrix factorization, where there exist only two entities $\{a, c\}$ and one relationship, the diagonal elements of Fisher information matrix is computed as

$$\mathbb{E}_{X,U}\left\{-\frac{\partial^2 \log p(\mathcal{X},\mathcal{U})}{\partial u_{i_a^* d^*}^{(a)} \partial u_{i_a^* d^*}^{(a)}}\right\} = \frac{N_{i_a^*}^{(a,c)} \rho_{d^*}^{(c)}}{\rho^{(a,c)}} + \frac{1}{\rho_{d^*}^{(a)}},$$

which is obviously smaller than the Fisher information matrix of the matrix co-factorizations. Exploiting additional matrices in the co-factorization model increases the Fisher information matrix as the number of observed entries grows larger, which lowers the CRB (the inverse of the Fisher information matrix).

3.2 Computing Reconstruction Error

The major difficulty regarding BCRB in matrix factorization is the non-uniqueness of the matrix decomposition. Instead of directly using the BCRB, we consider the reconstruction error $\mathcal{E}_{ij}$, which is written as

$$\begin{aligned}\mathcal{E}_{ij} &= \mathbb{E}\left\{(x_{ij} - \hat{x}_{ij})^2\right\}\\ &= \mathbb{E}\left\{(\boldsymbol{u}_i^\top \boldsymbol{v}_j - \hat{\boldsymbol{u}}_i^\top \hat{\boldsymbol{v}}_j)^2\right\},\end{aligned}$$

where $\hat{x}_{ij}$, $\hat{\boldsymbol{u}}_i$, and $\hat{\boldsymbol{v}}_j$ are the ground-truth values, and x_{ij} is the predicted value from the estimated parameters $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$. Although the matrix decomposition is not uniquely determined, the reconstruction error is the same for the decompositions having the same likelihood. The reconstruction error is lower-bounded by using the BCRB, in a way that

$$\begin{aligned}\mathcal{E}_{ij} &= \mathbb{E}\left\{(\boldsymbol{u}_i^\top \boldsymbol{v}_j - \hat{\boldsymbol{u}}_i^\top \boldsymbol{v}_j + \hat{\boldsymbol{u}}_i^\top \boldsymbol{v}_j - \hat{\boldsymbol{u}}_i^\top \hat{\boldsymbol{v}}_j)^2\right\}\\ &= \mathbb{E}\left\{\boldsymbol{v}_j^\top (\boldsymbol{u}_i - \hat{\boldsymbol{u}}_i^\top)(\boldsymbol{u}_i - \hat{\boldsymbol{u}}_i^\top)^\top \boldsymbol{v}_j\right\} + \mathbb{E}\left\{\hat{\boldsymbol{u}}_i^\top (\boldsymbol{v}_j - \hat{\boldsymbol{v}}_j)(\boldsymbol{v}_j - \hat{\boldsymbol{v}}_j)^\top \hat{\boldsymbol{u}}_i\right\}\\ &\quad +2\mathbb{E}\left\{\boldsymbol{v}_j^\top (\boldsymbol{u}_i - \hat{\boldsymbol{u}}_i)(\boldsymbol{v}_j - \hat{\boldsymbol{v}}_j)^\top \hat{\boldsymbol{u}}_i^\top\right\}\\ &\geq \mathbb{E}\left\{\boldsymbol{v}_j^\top \left[\mathcal{F}^{-1}\right]_{u_i} \boldsymbol{v}_j\right\} + \hat{\boldsymbol{u}}_i^\top \left[\mathcal{F}^{-1}\right]_{v_j} \hat{\boldsymbol{u}}_i + 2\mathbb{E}\left\{\boldsymbol{v}_j^\top (\boldsymbol{u}_i - \hat{\boldsymbol{u}}_i)(\boldsymbol{v}_j - \hat{\boldsymbol{v}}_j)^\top \hat{\boldsymbol{u}}_i^\top\right\}\\ &= \hat{\boldsymbol{v}}_j^\top \left[\mathcal{F}^{-1}\right]_{u_i} \hat{\boldsymbol{v}}_j + \operatorname{tr}\left(\left[\mathcal{F}^{-1}\right]_{u_i} \left[\mathcal{F}^{-1}\right]_{v_j}\right) + \hat{\boldsymbol{u}}_i^\top \left[\mathcal{F}^{-1}\right]_{v_j} \hat{\boldsymbol{u}}_i,\end{aligned}$$

where $\left[\mathcal{F}^{-1}\right]_{u_i}$ represents the part of the inverse of the Fisher information matrix corresponding to the parameter $\boldsymbol{u}_i$, which is a diagonal matrix whose elements consists of the negative second derivatives of the joint probability with respect to $\boldsymbol{u}_i$.

4 Numerical Experiments

We performed two experiments with BMCF. First experiment computed the BCRB for the matrix co-factorization model and matrix factorization model, and compared them with the actual performance of the BMCF and BMF algorithms. Second experiment ran the BMCF and BMF algorithm for the collaborative prediction problem, where the number of given ratings were adjusted to simulate the cold-start situations.

4.1 BCRB Comparison on Synthetic Data

For the experiment comparing the reconstruction error computed from BCRB and the actual performance of the algorithm, we generated synthetic data with four entities $\mathcal{E} = \{1, 2, 3, 4\}$ and three relationships $\mathcal{R} = \{(1, 2), (2, 3), (3, 4)\}$. The ground-truth factor matrices $\boldsymbol{U}^{(a)} \in \mathbb{R}^{5\times 100}$ were generated from the Gaussian distribution with variance 1. The relationship matrices were built from the factor matrices with additional Gaussian noise with variance 0.01. We chose the relation $(2, 3)$ as the target matrix, where half of columns have 50% of observed entries, and the remaining columns have varying ratio of observed entries from 0% to 90%. The other relation matrices had 50% of observed entries. To show the benefit of the co-factorization, we compared the BCRB of the matrix co-factorization model with BCRB of matrix factorization model which used the target relationship matrix only. The actual performance was measured using the proposed BMCF algorithm and BMF algorithm. We used the Root Mean Squared Error (RMSE) for the performance measure, which is computed by

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} |r_i - \overline{r}_i|^2},$$

where r_i represents the predicted value for the i-th test rating, $\overline{r}_i$ represents the true value, and N is the total number of test data points. Fig. 3 summarizes the result of the experiments. RMSE got better as the number of given ratings increases, both for the BCRB and the actual performance of the algorithm. BMCF had lower bound and performance compared to the BMF, and in this case the performance of BMCF was even lower than the theoretical lower-bound of BMF.

4.2 Collaborative Prediction in the Cold-Start Situation

We applied the proposed BMCF for the collaborative prediction problem in the cold-start situations, and compared the performance with that of the BMF to show the benefit of the BMCF. We used MovieLens data, which consists of the ratings of 943 users for the 1682 movies for the test. The ratings are given by the integer score from 1 to 5. MovieLens data is packed with the additional user and movie information, which were used in the matrix co-factorization.

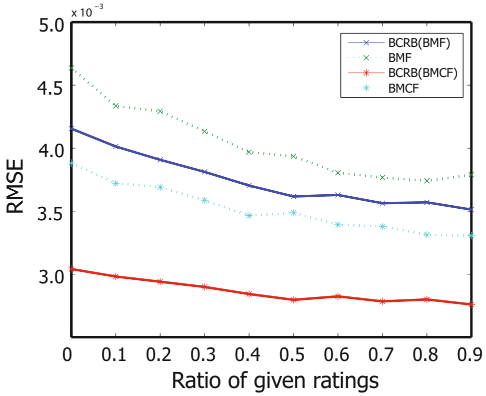


Fig. 3. Comparison of the BCRB and the performance of the BMCF and BMF, averaged over 10 different trials

We constructed the additional information matrices of users and items in the following manner. User information consists of the age, gender, and occupation. The ages are partitioned into 5 groups, which are: under 20, 21 to 30, 31 to 40, 41 to 50, and over 51. The corresponding entry for the user was marked as the indicating value 1. The gender and occupations were coded in the similar way, indicating the user's gender and occupations by using the value 1. Movie information, which consists of the 18 category of the movie genres, was also marked in the similar way. In the experiments, we used the user information matrix and the item information matrix, as well as the user-movie rating matrix.

To simulate the cold-start situations for the users, we randomly chose 200 users in the dataset for the test users and generated the training data with different number of given ratings. Along with RMSE, we also computed the Mean Absolute Error (MAE) which is computed by

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |r_i - \overline{r}_i|.$$

For each case, we randomly generated 10 different datasets, and ran the algorithm 10 times for each dataset with different initial values, so performance was measured 100 times for each case. Table 2(a) summarizes the averaged results for the experiments. BMF failed to predict the ratings when the test users have no ratings at all, however BMCF predicted fairly meaningful ratings for the case. The performance got better and better as the number of given ratings increases, but in all the cases, BMCF showed better performance than BMF, which showed the benefit of using side-information.

Another experiment was performed for the cases where some movies does not have any ratings at all. We randomly selected 100 movies from the dataset and eliminate all the ratings given for the movies. The averaged MAE and RMSE are summarized in Table 2(b). In this more severe condition, the performance of BMF

Table 2. Average MAE and RMSE results for different number of given ratings for each test user. (a) Simulation of user cold-start case. (b) Simulation of user and item cold-start case. We eliminate all ratings for 100 randomly chosen items to simulate item cold-start case.

(a)	BMF		BMCF		(b)	BMF		BMCF	
	MAE	RMSE	MAE	RMSE		MAE	RMSE	MAE	RMSE
0	2.5403	2.7767	0.8238	1.0140	0	2.5098	2.7584	0.8843	1.0857
5	0.8281	1.0618	0.7895	0.9941	5	0.9333	1.2412	0.8332	1.0550
10	0.8032	1.0205	0.7446	0.9424	10	0.8956	1.1863	0.7778	0.9857
15	0.7474	0.9558	0.7426	0.9314	15	0.8991	1.1948	0.7716	0.9789
20	0.7421	0.9496	0.7348	0.9254	20	0.8618	1.1535	0.7527	0.9555

was seriously degraded from the performance for the previous experiment. However, BMCF showed much better performance than BMF for all cases, slightly less than the results of the previous experiment. The use of the additional item information by using BMCF greatly helped the performance of the prediction, especially in this kind of item cold-start (as well as user cold-start) cases.

5 Conclusions

We have presented Bayesian matrix co-factorization (BMCF) as an approach to incorporating side information into collaborative prediction, where multiple data matrices are jointly decomposed, with some factor matrices shared over inter-related factorizations, in Bayesian setting. We have presented variational inference algorithm for updating factor matrices, in which variational posterior means and variances for factor matrices are iteratively updated. Hyperparameters are determined by maximizing the marginal likelihood. We have calculated Bayesian Cramér-Rao bound for the matrix co-factorization model, stressing that the co-factorization actually lowers the theoretical bound of the reconstruction error. Numerical experiments demonstrated that Bayesian matrix co-factorization yielded the lower BCRB and improved the performance in collaborative prediction, compared to Bayesian matrix factorization. Especially in the case of cold start problems, Bayesian matrix co-factorization led to the satisfactory performance, while Bayesian matrix factorization failed to make proper predictions.

Acknowledgments. This work was supported by National Research Foundation (NRF) of Korea (2010-0014306, 2010-0018828, 2010-0018829), NIPA Program of Software Engineering Technologies Development and Experts Education, and NRF WCU Program (R31-10100).

References

1. Kay, S.M.: Fundamentals of Statistical Signal Processing: Estimation Theory. Prentice-Hall, Englewood Cliffs (1993)
2. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer 42(8), 30–37 (2009)

3. Lee, H., Choi, S.: Group nonnegative matrix factorization for EEG classification. In: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), Clearwater Beach, Florida (2009)
4. Lim, Y.J., Teh, Y.W.: Variational Bayesian approach to movie rating prediction. In: Proceedings of KDD Cup and Workshop, San Jose, CA (2007)
5. Raiko, T., Ilin, A., Karhunen, J.: Principal component analysis for large scale problems with lots of missing values. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 691–698. Springer, Heidelberg (2007)
6. Rennie, J.D.M., Srebro, N.: Fast maximum margin matrix factorization for collaborative prediction. In: Proceedings of the International Conference on Machine Learning (ICML), Bonn, Germany (2005)
7. Salakhutdinov, R., Mnih, A.: Bayesian probablistic matrix factorization using MCMC. In: Proceedings of the International Conference on Machine Learning (ICML), Helsinki, Finland (2008)
8. Salakhutdinov, R., Mnih, A.: Probablistic matrix factorization. In: Advances in Neural Information Processing Systems (NIPS), vol. 20. MIT Press, Cambridge (2008)
9. Singh, A.P., Gordon, G.J.: Relational learning via collective matrix factorization. In: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Las Vegas, Nevada (2008)
10. Singh, A.P., Gordon, G.J.: A Bayesian matrix factorization model for relational data. In: Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI), Catalina Island, CA (2010)
11. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Scalable collaborative filtering approaches for large recommender systems. Journal of Machine Learning Research 10, 623–656 (2009)
12. Tichavsky, P., Muravchik, C.H., Nehorai, A.: Posterior Cramér-Rao bounds for discrete-time nonlinear filtering. IEEE Transactions on Signal Processing 46(5), 1386–1395 (1998)
13. Williamson, S., Ghahramani, Z.: Probabilistic models for data combination in recommender systems. In: NIPS 2008 Workshop on Learning from Multiple Sources, Whistler, Canada (2010)
14. Yoo, J., Choi, S.: Weighted nonnegative matrix co-tri-factorization for collaborative prediction. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS, vol. 5828, pp. 396–411. Springer, Heidelberg (2009)
15. Yoo, J., Choi, S.: Matrix co-factorization on compressed sensing. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Barcelona, Spain (2011)
16. Yu, K., Yu, S., Tresp, V.: Multi-label informed latent semantic indexing. In: Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Salvador, Brazil (2005)
17. Zhu, S., Yu, K., Chi, Y., Gong, Y.: Combining content and link for classification using matrix factorization. In: Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Amsterdam, The Netherlands (2007)

Learning from Inconsistent and Unreliable Annotators by a Gaussian Mixture Model and Bayesian Information Criterion

Ping Zhang and Zoran Obradovic

Center for Data Analytics and Biomedical Informatics, Temple University,
Philadelphia, PA 19122, USA
{ping,zoran.obradovic}@temple.edu

Abstract. Supervised learning from multiple annotators is an increasingly important problem in machine leaning and data mining. This paper develops a probabilistic approach to this problem when annotators are not only unreliable, but also have varying performance depending on the data. The proposed approach uses a Gaussian mixture model (GMM) and Bayesian information criterion (BIC) to find the fittest model to approximate the distribution of the instances. Then the maximum a posterior (MAP) estimation of the hidden true labels and the maximum-likelihood (ML) estimation of quality of multiple annotators are provided alternately. Experiments on emotional speech classification and CASP9 protein disorder prediction tasks show performance improvement of the proposed approach as compared to the majority voting baseline and a previous data-independent approach. Moreover, the approach also provides more accurate estimates of individual annotators performance for each Gaussian component, thus paving the way for understanding the behaviors of each annotator.

Keywords: multiple noisy experts, data-dependent experts, Gaussian mixture model, Bayesian information criterion.

1 Introduction

In supervised learning, it is usually assumed that true labels are readily available from a single annotator or source. However, recent advances in corroborative technology have given rise to situations where the true label of the target is unknown. In such problems, multiple sources or annotators are often available that provide noisy labels of the targets. For example, in the area of computer-aided diagnosis (CAD) the actual gold standard (whether the suspicious region is malignant or not) can only be obtained from a biopsy of the tissue. Since it is an expensive, invasive, and potentially dangerous process, often CAD systems are built from labels assigned by multiple radiologists who provide subjective and possibly noisy version of the gold standard. Very often there is a lot of disagreement among the labels. Another example is Amazon Mechanical Turk (AMT) [1] which allows the requesters to publish any Human

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 553–568, 2011.

Intelligence Tasks (HIT) on the website, such as writing essays, filling out certain questionnaires, or just collecting and labeling data. Any user of AMT can finish the tasks he is interested in and get paid. Therefore, acquiring non-expert labels is now easy, fast and inexpensive. On the other hand, since there is little control for the annotators, there is no guarantee for labeling quality: there could be careless, fallible, irresponsible or even malicious annotators.

In these multi-annotator problems, building a classifier in the traditional single annotator manner, without regard for the annotator properties may not be effective in general. The reasons for this include: some annotators may be more reliable than others, some may be malicious, some may be correlated with others, and in particular annotator effectiveness may vary depending on the data instance presented. In recent years, how to make the best use of the labeling information provided by multiple annotators to approximate the hidden true concept has drawn the attention of researchers in machine learning and data mining.

There has already been some literature for dealing with the multi-annotator setting. One popular strategy is to assign each sample to multiple annotators for labeling [2-6]. This repeated labeling strategy relies on the identification of what labels should be reacquired in order to improve classification performance or data quality. This form of active learning can be well suited when we can control assignments of samples to labelers. However, there are many cases that we have no access in doing so. Even we have, getting multiple labels for one sample could be a great waste of resources. As a result, research is conducted on the methods without using repeated labeling. These include techniques where labeler similarities are used to identify what samples should be used to estimate classification models for each labeler [7], and where low-quality annotators are pruned out by using the model trained from the entire dataset with all annotators as a ground truth [8]. Application areas for multi-annotator learning vary widely. These include natural language processing [9], computer-aided diagnosis [10, 11], computer vision [12, 13], speech technology [14] and bioinformatics [15, 16].

Among these papers, an elegant probabilistic framework of iteratively evaluating the different annotators and giving an estimate of the hidden true labels is developed [11]. However, the approach assumes the error rate of each annotator is consistent across all the input data. This is an impractical assumption in many cases since annotator knowledge can fluctuate considerably depending on the groups of input instances. For example, radiologists specialized in heart images will be better at labeling lesions of the heart compared to radiologists with lung expertise, who on the other hand would label instances of lung diseases better. In this paper, our proposed approach follows prior work [11] but relaxes the data-independent assumption, i.e., we assume an annotator may not be consistently accurate across the entire feature space. A very recent paper [17] also developed a data-dependent probabilistic model by assuming each annotator provides a Bernoulli noisy version or Gaussian distorted version of the true label. Compared to [17] our proposed approach first uses GMM and BIC to find a fittest model to approximate the distribution of the instances. Then, it alternately provides the maximum a posterior (MAP) estimation of the hidden true labels and the maximum-likelihood (ML) estimation of quality of multiple annotators at each mixture component.

The remaining part of this paper consists of the background on GMM and BIC, followed by a description of the approach, the summary of experimental results, conclusions and discussions of future work.

2 Background

In this section, we introduce the way of using GMM to approximate the distribution of the instances and using BIC to find the fittest GMM, and summarize the background information into Algorithm 1 and Algorithm 2. Our proposed approach (in Section 3) used these two algorithms to build a data-dependent probabilistic model of multiple noisy annotators.

2.1 The Gaussian Mixture Model

Given observations $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$, in a Gaussian mixture each component is modeled by a multivariate normal distribution. The parameters of component k comprise the mean vector μ_k, the covariance matrix Σ_k, and the probability density function

$$f_k(\boldsymbol{x}_i \mid \mu_k, \Sigma_k) = \frac{\exp\{-\frac{1}{2}(\boldsymbol{x}_i - \mu_k)^T \Sigma_k^{-1} (\boldsymbol{x}_i - \mu_k)\}}{(2\pi)^{d/2} \mid \Sigma_k \mid^{1/2}} .$$

Let K be the number of components in the mixture, and let π_k be mixing proportions: $0 < \pi_k < 1, \sum_{k=1}^{K} \pi_k = 1$. We wish to estimate the parameters $\theta = \pi_1, ..., \pi_K, \mu_1, ..., \mu_K, \Sigma_1, ..., \Sigma_K$. Then the log likelihood of the mixture is

$$L(\theta \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_N) = \sum_{i=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}_i \mid \mu_k, \Sigma_k)\} . \tag{1}$$

Here, Σ_k determines the geometric properties of component k. In [18] a general framework is proposed for exploiting the representation of the covariance matrix in terms of its eigenvalue decomposition $\Sigma_k = \lambda_k D_k A_k D_k^T$, where D_k is the orthogonal matrix of eigenvectors, A_k is a diagonal matrix whose elements are proportional to the eigenvalues of Σ_k , and λ_k is a scalar. The matrix D_k determines the orientation of the component, A_k determines its shape, and λ_k determines its volume. Allowing some but not all of the parameters in the decomposition to vary results in a set of models within this general framework. Such an approach is sufficiently flexible to accommodate data with widely varying characteristics. In this paper, by following the discussion of [19] we used 9 parameterizations which have a closed form update for the covariance matrix.

To estimate the parameters of the Gaussian mixture we used the Expectation-Maximization (EM) algorithm [20] which is introduced in Algorithm 1.

Algorithm 1 (EM for Gaussian mixtures).

Input: Observed data $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$, the number of Gaussian components K, and the form of the covariance matrix.

Output: The model parameters $\theta = \pi_1, ..., \pi_K, \mu_1, ..., \mu_K, \Sigma_1, ..., \Sigma_K$, the responsibilities τ_{ik} which are the probabilities that $\boldsymbol{x}_i$ is generated by component k, $i = 1, ..., N, k = 1, ..., K$

Step 1. Use K-means algorithm to initialize the means μ_k, covariances Σ_k and mixing coefficients π_k, and evaluate the initial value of the log likelihood by equation (1).

Step 2. (E-step) Evaluate the responsibility of the current model parameters, i.e. the probability that an observation $\boldsymbol{x}_i$ belongs to component k as

$$\tau_{ik} = \frac{\pi_k f_k(\boldsymbol{x}_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j f_j(\boldsymbol{x}_i \mid \mu_j, \Sigma_j)}$$

Step 3. (M-step) Re-estimate the parameters using the current estimated probability as follows

$$\pi_k^{new} = \frac{1}{N} \sum_{i=1}^{N} \tau_{ik}$$

$$\mu_k^{new} = \frac{1}{N} \sum_{i=1}^{N} \frac{\tau_{ik} \boldsymbol{x}_i}{\pi_k^{new}}$$

$$\Sigma_k^{new} = \frac{1}{N} \sum_{i=1}^{N} \frac{\tau_{ik} (\boldsymbol{x}_i - \mu_k^{new})(\boldsymbol{x}_i - \mu_k^{new})^T}{\pi_k^{new}} \quad ^1$$

Step 4. Evaluate the log likelihood by (1) using the updated parameters and check for convergence of either parameters or the log likelihood. If the convergence criterion is not satisfied return to Step 2.

2.2 Bayesian Model Selection

Each combination of a different specification of covariance matrices and a different number of components corresponds to a separate probability model. One advantage of the Gaussian mixture model is that it allows the use of approximate Bayes factors to compare models. This gives a systematic means of selecting not only the parameterization of the model, but also the number of components.

Let X be the observed data, M_1 and M_2 be two models with parameters θ_1 and θ_2 respectively. The integrated likelihood is defined as $p(X \mid M_g) =$

[1] The update for covariance matrix is only for the unconstrained case. See [19] for a complete description of the update equations for all 9 models.

$\int p(X \mid \theta_g, M_g) p(\theta_g \mid M_g) d\theta_g$ where $g = 1,2$ and $p(\theta_g \mid M_g)$ is the prior distribution of θ_g. The integrated likelihood represents the probability that data X is observed given that the underlying model is M_g. The Bayes factor is defined as the ratio of the integrated likelihoods of the two models, i.e. $B_{12} = P(X \mid M_1) / P(X \mid M_2)$. In other words, the Bayes factor B_{12} represents the posterior odds that the data were distributed according to M_1 against model M_2 assuming that neither model is favored a priori. If $B_{12} > 1$, model M_1 is favored over M_2. The method can be generalized to more than two models. The main difficulty in using the Bayes factor is the evaluation of the integrated likelihood. By following the discussion of [21], we used an approximation called the Bayesian Information Criterion (BIC), given by

$$p(X \mid M_g) \approx BIC_g = 2\log(X \mid \widehat{\theta_g}, M_g) - m_g \log(N) \tag{2}$$

where m_g is the number of independent parameters that must be estimated for model M_g, and $\widehat{\theta_g}$ is the maximum-likelihood estimate for parameter θ_g. A large BIC score indicates strong evidence for the corresponding model. Hence, the BIC score can be used to compare models with different covariance matrix parameterizations and different numbers of components.

Here, we summarize the procedure of selecting the best Gaussian mixture model in Algorithm 2.

Algorithm 2 (Bayesian Model Selection for Gaussian mixtures).

Input: Observed data $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$.

Output: The optimal number of components, the optimal form of the component densities, and the corresponding model parameters and components responsibilities for each instance.

Step 1. Choose a form of model M from the 9 candidate models [19].

Step 2. Choose a number of components k. Here check from 1 to 6 (the maximum number of components).

Step 3. Use Algorithm 1 to obtain model parameters and log likelihood for this M and k.

Step 4. Calculate the value of BIC for this M and k by using equation (2).

Step 5. Go to Step 2 to choose another value of k.

Step 6. Go to Step 1 to choose another form of model M.

Step 7. Choose the optimal configuration (number of components and form of the covariance matrices) that corresponds to the highest BIC.

3 Method

Given a dataset $D = \{\boldsymbol{x}_i, y_i^1, ..., y_i^R\}_{i=1}^N$ containing N instances, where $\boldsymbol{x}_i$ is an instance (typically a d-dimensional feature vector), $y_i^j \in \{0,1\}$ is the corresponding binary label assigned to the instance $\boldsymbol{x}_i$ by the j-th annotator and R is the number of annotators.

Based on the intuition that real world annotators have different sensitivity and specificity for different regions of the entire feature space, we introduced a new data-dependent model in this paper. By using Algorithm 2, a fittest K-mixture-component GMM is used to approximate the distribution of the instances. To model the data-dependent behavior of annotators, we hypothesize that each annotator has its own sensitivity and specificity for each mixture component. The sensitivity α_k^j and specificity β_k^j are defined as follows:

$$\alpha_k^j = \Pr(y_i^j = 1 \mid y_i = 1, \text{k-th Gaussian mixture component generates } \boldsymbol{x}_i) \quad (3)$$

$$\beta_k^j = \Pr(y_i^j = 0 \mid y_i = 0, \text{k-th Gaussian mixture component generates } \boldsymbol{x}_i) \quad (4)$$

where $j = 1,...,R$; $k = 1,...,K$. We hypothesize that annotators generate labels as follows: given an instance $\boldsymbol{x}_i$ to label, the annotators find the mixture component which most possible generates that instance. Then the annotators generate labels with their sensitivities and specificities at that most possible component.

Our task is not only to get an estimation of the unknown true labels $y_1,..., y_N$, but also to estimate the sensitivity (i.e. true positive rate) $\boldsymbol{\alpha} = [\alpha_1^1,...,\alpha_k^j,...,\alpha_K^R]$ and the specificity (i.e. true negative rate) $\boldsymbol{\beta} = [\beta_1^1,...,\beta_k^j,...,\beta_K^R]$ of the R annotators at K Gaussian mixture components.

To fulfill the task defined before, we propose an iterative algorithm that we will call GMM-MAPML. Given dataset D , we use Algorithm 2 to get parameters of the fittest GMM and its mixture components' responsibilities for each instance. Also, we use majority voting to initialize the probabilistic labels z_i (i.e., the probability when the hidden true label is 1). Then, the algorithm alternately carries out the ML estimation and the MAP estimation which described in details in the following subsections. Given the current estimates of probabilistic labels z_i , the ML estimation measures annotators' performance (i.e., their sensitivity $\boldsymbol{\alpha}$ and specificity $\boldsymbol{\beta}$) at each mixture component and learns a classifier with parameter $\boldsymbol{w}$. Given the estimated sensitivity $\boldsymbol{\alpha}$, specificity $\boldsymbol{\beta}$, and the prior probability which is provided by the learned classifier, the MAP estimation gets the updated probabilistic labels z_i based on the Bayesian rule. After the two estimations converge, we get the algorithm outputs which include both the probabilistic labels z_i and the model parameters $\phi = \{\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$.

3.1 ML Estimation of the Model Parameters

Given a dataset D and the current estimates of z_i , we estimate the model parameters $\phi = \{\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$ by maximizing the conditional likelihood.

Denote $z_{ik} = z_i \tau_{ik}$, where τ_{ik} is the probability that $\boldsymbol{x}_i$ is generated by component k, according to (3) and (4) we can get the sensitivity of j-th annotator at k-th component and the specificity of j-th annotator at k-th component as

$$\alpha_k^j = \sum_{i=1}^{N} z_{ik} y_i^j \Big/ \sum_{i=1}^{N} z_{ik}$$
$$\beta_k^j = \sum_{i=1}^{N} (\tau_{ik} - z_{ik})(1 - y_i^j) \Big/ \sum_{i=1}^{N} (\tau_{ik} - z_{ik}) \tag{5}$$

Given probabilistic labels z_i, we can learn any classifier using ML estimation. However, in this section for convenience, we will explain it with a logistic regression classifier. By using that classifier, the probability for the positive class is modeled as a sigmoid acting on the linear discriminating function, that is,

$$\Pr[y = 1 \mid \boldsymbol{x}, \boldsymbol{w}] = \sigma(\boldsymbol{w}^T \boldsymbol{x}) \tag{6}$$

where the logistic sigmoid function is defined as $\sigma(x) = 1/(1 + e^{-x})$. To estimate the classifier's parameter $\boldsymbol{w}$, we use a gradient descent method, that is, the Newton-Raphson method [22]

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta \boldsymbol{H}^{-1} \boldsymbol{g} \tag{7}$$

where **g** is the gradient vector, **H** is the Hessian matrix, and η is the step length. The gradient vector is given by $\boldsymbol{g}(\boldsymbol{w}) = \sum_{i=1}^{N} [z_i - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)] \boldsymbol{x}_i$, and the Hessian matrix is given by $\boldsymbol{H}(\boldsymbol{w}) = -\sum_{i=1}^{N} [\sigma(\boldsymbol{w}^T \boldsymbol{x}_i)][1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)] \boldsymbol{x}_i \boldsymbol{x}_i^T$.

3.2 MAP Estimation of the Unknown True Labels

Given a dataset D and the model parameters $\phi = \{\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$, the probabilistic labels are $z_i = \Pr[y_i = 1 \mid y_i^1, ..., y_i^R, \boldsymbol{x}_i, \phi]$. Using the Bayesian rule we have

$$z_i = \frac{\Pr[y_i^1, ..., y_i^R \mid y_i = 1, \phi] \cdot \Pr[y_i = 1 \mid \boldsymbol{x}_i, \phi]}{\Pr[y_i^1, ..., y_i^R \mid \phi]} \tag{8}$$

which is a MAP estimation problem.

Conditioning on the true label $y_i \in \{1, 0\}$, the denominator of formula (8) is decomposed as

$$\begin{aligned} &\Pr[y_i^1, ..., y_i^R \mid \phi] = \\ &\Pr[y_i^1, ..., y_i^R \mid y_i = 1, \boldsymbol{\alpha}] \Pr[y_i = 1 \mid \boldsymbol{x}_i, \boldsymbol{w}] \\ &+ \Pr[y_i^1, ..., y_i^M \mid y_i = 0, \boldsymbol{\beta}] \Pr[y_i = 0 \mid \boldsymbol{x}_i, \boldsymbol{w}] \end{aligned} \tag{9}$$

In our data-dependent model, given an instance $\boldsymbol{x}_i$ to label, the j-th annotator finds the q-th mixture component which most possible generates that instance. Then the annotator generates a label with the sensitivity α_q^j and specificity β_q^j. Therefore,

$$\Pr[y_i^1,...,y_i^R \mid y_i = 1, \boldsymbol{\alpha}] = \Pr[y_i^1,...,y_i^R \mid y_i = 1, \alpha_q^1,...,\alpha_q^R] \tag{10}$$

where $q = \arg\max_{k=1,...,K}(\tau_{ik})$. At each component, given the true label y_i we assume that $y_i^1,...,y_i^R$ are independent, that is, the annotators label the instances independently. Hence,

$$\begin{aligned}\Pr[y_i^1,...,y_i^R \mid y_i = 1, \alpha_q^1,...,\alpha_q^R] &= \prod_{j=1}^{R} \Pr[y_i^j \mid y_i = 1, \alpha_q^j] \\ &= \prod_{j=1}^{R} [\alpha_q^j]^{y_i^j} [1-\alpha_q^j]^{1-y_i^j}\end{aligned} \tag{11}$$

Similarly, we have

$$\Pr[y_i^1,...,y_i^R \mid y_i = 0, \boldsymbol{\beta}] = \prod_{j=1}^{R} [1-\beta_q^j]^{y_i^j} [\beta_q^j]^{1-y_i^j} \tag{12}$$

From (6), (8), (9), (10), (11) and (12), the posterior probability z_i which is a soft probabilistic estimate of the hidden true label is computed as

$$z_i = \frac{a_i p_i}{a_i p_i + b_i(1-p_i)} \tag{13}$$

where

$$\begin{aligned} p_i &= \Pr[y_i = 1 \mid \boldsymbol{x}_i, \boldsymbol{w}] = \sigma(\boldsymbol{w}^T \boldsymbol{x}_i) \\ a_i &= \prod_{j=1}^{R} [\alpha_q^j]^{y_i^j} [1-\alpha_q^j]^{1-y_i^j} \\ b_i &= \prod_{j=1}^{R} [1-\beta_q^j]^{y_i^j} [\beta_q^j]^{1-y_i^j} \\ q &= \arg\max_{k=1,...,K}(\tau_{ik}) \end{aligned}.$$

3.3 The GMM-MAPML Algorithm

We summarize the iterative approach in Algorithm 3.

Algorithm 3 (Iterative GMM-MAPML Algorithm).

Input: Dataset $D = \{\boldsymbol{x}_i, y_i^1,..., y_i^R\}_{i=1}^N$ containing N instances. Each instance has binary labels from R annotators.

Output: The fittest K-mixture-component GMM for the instances; the estimated sensitivity and specificity of each annotator at each mixture component; the weight parameter of a classifier; the probabilistic labels z_i; the estimation of the hidden true label y_i.

Step 1. Find the fittest K-mixture-component GMM for the instances, and get the corresponding GMM parameters and components responsibilities for each instance by Algorithm 2.

Step 2. Use majority voting to initialize $z_i = \sum_{j=1}^{R} y_i^j \Big/ R$.

Step 3. Iterative optimization.

(a) ML estimation – Estimate the model parameters $\phi = \{\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$ based on current probabilistic labels z_i using (5) and (7).

(b) MAP estimation – Given the model parameters ϕ, update z_i using (13).

Step 4. If ϕ and z_i do not change between two successive iterations or the maximum number of iterations is reached, go to the Step 5; otherwise, go back to the Step 3.

Step 5. Estimate the hidden true label y_i by applying a threshold γ on z_i, that is, y_i =1 if $z_i > \gamma$ and y_i =0 otherwise.

3.4 Analysis of the Model

To explain how the model works, we apply the logit function to the posterior probability z_i. From equation (13), the logit of z_i is written as

$$\begin{aligned} \text{logit}(z_i) &= \ln \frac{z_i}{1-z_i} = \ln \frac{\Pr[y_i = 1 \mid y_i^1, \dots, y_i^R, \boldsymbol{x}_i, \phi]}{\Pr[y_i = 0 \mid y_i^1, \dots, y_i^R, \boldsymbol{x}_i, \phi]} \\ &= w^T \boldsymbol{x}_i + \sum_{j=1}^{R} y_i^j [\text{logit}(\alpha_q^j) + \text{logit}(\beta_q^j)] + c \end{aligned} \tag{14}$$

where $c = \sum_{j=1}^{R} \ln[(1-\alpha_q^j) \big/ \beta_q^j]$, and $q = \arg\max_{k=1,\dots,K}(\tau_{ik})$. The first term of (14) is a linear combination (provided by the learned classifier) of features of instance $\boldsymbol{x}_i$. The second term of (14) is a weighted linear combination of the labels from all annotators. The weight of each annotator is the sum of the logit of the estimated sensitivity and specificity at the q-th component, i.e., the most possible component for generating $\boldsymbol{x}_i$. Therefore, our proposed model is data-dependent. Also, from equation (14) we can infer that the estimates of the hidden true labels depend both on observations and on the labels from all annotators.

4 Experimental Results

In this section we experimentally validate the proposed approach on two real-life datasets.

4.1 Emotional Speech Classification Experiment

Emotion recognition is an area which attracts interest from the speech research community. A wide area of applications such as interface optimization and expressive voice synthesis are related to the classification of speech into emotional states. In this experiment, we used a publicly available dataset from the EMA database [23]. This dataset has 3 speakers: a male native speaker read 14 sentences, and two female native speakers read 10 sentences. Each sentence was produced five times for four acted emotions, i.e., neutral, angry, sad, and happy. In this dataset, each utterance was evaluated by at least 3 expert listeners and 568 utterances were chosen as best emotion utterances. In our experiment, we used these 568 utterances as instances and their experts verified target emotions as the ground truth labels. Following the experiments in [14], {happy, neutral} were assigned to class 0 as positive emotion, and {sad, angry} were assigned to class 1 as negative emotion. To get the labels from multiple annotators, we sent the raw audio files (in WAV format) to 5 inexperienced listeners and asked them to provide binary labels (0 for positive emotion, 1 for negative emotion) for each instance. The 5 annotators have different academic backgrounds, and most of them are non-native speakers. By using VOICEBOX [24], we extracted 13 static features (12 MFCCs computed from 24 filter banks and log energy), 13 delta coefficients (first derivatives of static features) and 13 delta-delta coefficients (second derivatives of static features) from the speech signal over 25 ms frames with 10 ms overlap. The feature-wise mean is computed over the entire utterance, resulting in a 39-element feature vector for each instance.

In our comparisons, we considered three multiple-annotator methods: (1) Majority Voting that uses the average of annotators' votes as the estimation of the hidden true label; (2) MAP-ML that estimates the hidden true labels and annotators' constant accuracy across all the input data using a data-independent model [11, 16]; and (3) GMM-MAPML that uses our proposed data-dependent model as described in Section 3. For further comparisons, we also learned two additional logistic regression classifiers: (4) LR Concatenation that concatenates all annotators' labels as a training set, and (5) LR Ground Truth that uses the actual ground truth as a training set. For both LR Concatenation and LR Ground Truth, we randomly divided the whole dataset into five equally sized folds (20% of the dataset each). We repeated five times the logistic regression model training where we used four of the folds (80% of the dataset) for training and one fold for testing.

The ROC comparisons for three multiple-annotator methods and two additional logistic regression classifiers are shown in Fig. 1. The figure demonstrates the power of our proposed GMM-MAPML approach: GMM-MAPML significantly outperforms baseline methods (Majority Voting and MAP-ML) where information from all annotators is taken into account in a more naïve way. In addition, GMM-MAPML successfully approximates the LR Ground Truth classifier which is trained by the actual true labels. The Fig. 1 also shows that building a classifier in the traditional single annotator manner (simply concatenates all annotators' labels as LR Concatenation) without regard for the annotator properties may not be effective for the multi-annotator problems.

As an output of our proposed GMM-MAPML method, a two-Gaussian-component model with an unconstrained covariance matrix has been selected for the emotional speech data. For each component, estimated sensitivity and specificity of 5 listeners using GMM-MAPML are shown in Table 1. The table shows that the 5 listeners have different sensitivity and specificity at the two components. Taking a closer look at the emotional speech data, we found that 75.35% of the utterances produced by the male speaker have the first Gaussian component as their principle component (i.e., the most possible component) and 64.31% of the utterances produced by the female speakers have the second Gaussian component as their principle component. It seems like some listeners (e.g., Listener 2 and Listener 4 in our experiment) are good at labeling one gender's utterance, but not at other gender's. The analysis shows that our proposed GMM-MAPML approach can be used for selecting the best annotators for instances at different components (or in different regions) of the feature space and for the training of annotators by informing them about the set of examples which they labeled unreliably.

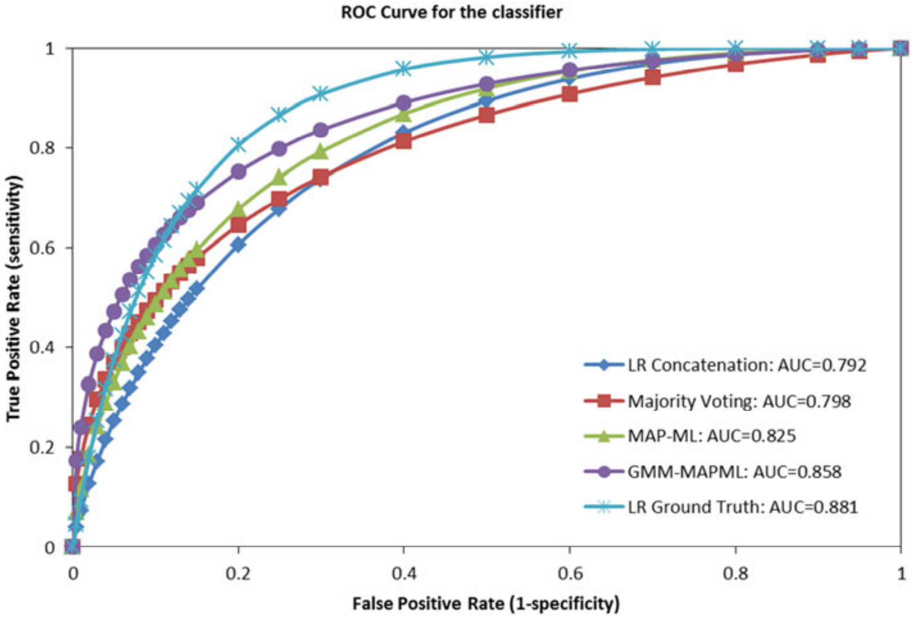


Fig. 1. The ROC comparisons for three multiple-annotator methods and two logistic regression classifiers on the emotional speech classification task. Methods are sorted in legend of the figure according to their AUC value.

Table 1. GMM-MAPML based estimates of 5 listeners' accuracy in emotional speech data (first and second component) without using golden ground truth

	First Component		Second Component	
Listeners	Estimated Sensitivity	Estimated Specificity	Estimated Sensitivity	Estimated Specificity
Listener 1	0.902	0.891	0.925	0.951
Listener 2	0.843	0.862	0.814	0.799
Listener 3	0.784	0.802	0.779	0.792
Listener 4	0.756	0.744	0.877	0.861
Listener 5	0.719	0.698	0.728	0.736

4.2 CASP9[2] Protein Disorder Prediction Experiment

Computational characterization of disorder in proteins is appealing due to the difficulties and high cost involved in experimental characterization of disorders. Treating an individual predictor as an annotator, the multiple-annotator methods can be used to build meta-predictors for protein disorder prediction. Recently, a data-independent model based on the idea of the MAP-ML method discussed in section 4.1 is used to integrate the prediction labels from multiple predictors [16]. This is shown to improve accuracy in performed experiments as compared to using individual component predictors. Following the experiments in [16], to characterize the method proposed in our study we used CASP9 data [25] consisting of 117 protein sequences with 26,083 amino-acid residues. For each residue, the golden ground truth (i.e. the residue is either in ordered state or in disordered state) was obtained by either X-ray or NMR experimental characterization. We have also obtained prediction labels (1 represents a disordered state while 0 represents an ordered state) with disorder probabilities (values in the range of 0–1) of all predictors which participated in CASP9 from the contest's official website [25]. We selected 15 predictors developed by groups at different institutions assuming that their errors are independent. By following the method proposed in [16], we extracted a 20-dimensional feature vector (19 amino acid composition features and 1 sequence complexity feature) for each residue.

In the experiment, as the input of our GMM-MAPML algorithm we used the 26,083 amino-acid instances and the prediction labels from the 15 individual predictors. After the algorithm had converged, we used the estimation of the hidden true labels y_i (given the threshold of 0.5) produced by GMM-MAPML as the binary disorder/order predictions and the probabilistic labels z_i from GMM-MAPML outputs as the disorder probability. As alternatives we also used the other two multiple-annotator methods, i.e., MAP-ML and Majority Voting to integrate the individual predictors, so that we can compare those methods with the GMM-MAPML to see which one is more effective. Following the regulation of CASPs, performance of the methods was evaluated by three criteria [26]: (1) the average of sensitivity and specificity (ACC), (2) a weighted score (S_w) that considers the rates of ordered and disordered residues in the data; (3) and the area under the ROC curve (AUC).

Comparisons of 15 individual predictors, the MAP-ML method, the Majority Voting method, and our GMM-MAPML method on CASP9 data is shown in Table 2. Our proposed GMM-MAPML method significantly outperforms the two baseline methods (Majority Voting and MAP-ML) and each individual predictor in all three criteria.

Using the BIC, our GMM-MAPML method also finds that the fittest GMM for CASP9 data is three Gaussian components with the covariance matrix in the form of $\lambda_k \boldsymbol{B}_k$ (**B** is a diagonal matrix). For each component, estimated sensitivity and specificity of 15 individual predictors using GMM-MAPML without relying on golden ground truth are shown in Fig. 2. The obtained estimates are sorted according to the average of their estimated sensitivity and specificity. The Fig. 2 clearly shows that the individual CASP9 disorder predictors have different sensitivity and specificity at

[2] CASP9 is the abbreviation of the 9th Biannual Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction held in year 2010.

different components. The figure also demonstrates that the rankings of individual predictors are different at different components.

For further analysis, the relationship between amino-acid residue positions and the three Gaussian components are shown in Fig. 3. We found that the principal (most likely) component at the N-terminus (defined here as 20% of residues at the start of a protein sequence) was the first Gaussian component. In particular, about 56% of the amino-acid residues from this region belong to this component. The principal component for the C-terminus consisting of 20% of residues at the end of each protein was the third component (59% of residues from this region belong to this component). The internal 60% of residues were most likely to belong to the second Gaussian component (54% of these residues belong to this component). The results well agree with previous protein disorder work where amino-acid residues at different regions (i.e., N-terminal, C-terminal and internal) have different compositions and different tendencies for disorder [27].

The experiment on CASP9 data shows that our proposed GMM-MAPML method can potentially be used to improve prediction of protein disorder and to provide helpful suggestion on choosing the suitable disorder predictors for each region of unknown protein sequences.

Table 2. Comparisons of GMM-MAPML vs. alternative protein disorder meta-predictors (MAP-ML and MAJORITY VOTING) and individual CASP9 predictors according to CASP9 evaluation measures using CASP9 data

Predictor Name	ACC	S_w	AUC
GMM-MAPML	0.785	0.527	0.874
MAP-ML	0.764	0.513	0.859
MAJORITY VOTING	0.735	0.496	0.776
PRDOS2	0.754	0.509	0.855
MULTICOM-REFINE	0.750	0.500	0.822
BIOMINE_DR_PDB	0.741	0.483	0.821
GSMETADISORDERMD	0.738	0.476	0.816
MASON	0.736	0.473	0.743
ZHOU-SPINE-D	0.731	0.462	0.832
DISTILL-PUNCH1	0.726	0.453	0.800
OND-CRF	0.706	0.412	0.737
UNITED3D	0.704	0.412	0.781
CBRC_POODLE	0.694	0.405	0.830
MCGUFFIN	0.688	0.402	0.817
ISUNSTRUCT	0.679	0.396	0.742
DISOPRED3C	0.670	0.391	0.853
ULG-GIGA	0.585	0.341	0.726
MEDOR	0.579	0.338	0.688

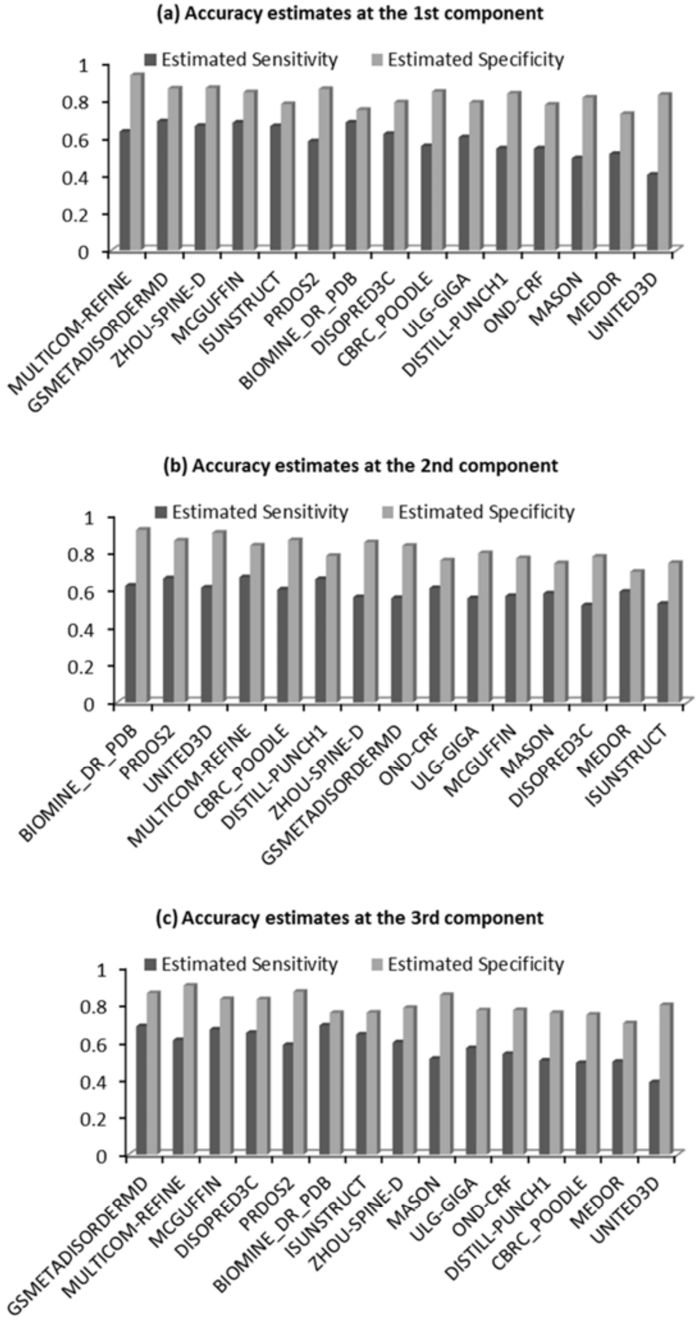


Fig. 2. Analysis of CASP9 disorder predictors at three principal components of CASP9 data identified by GMM-MAPML. In panels a, b, and c the predictors are sorted in descending order of the average of the estimated sensitivity and specificity on the corresponding component of CASP9 data.

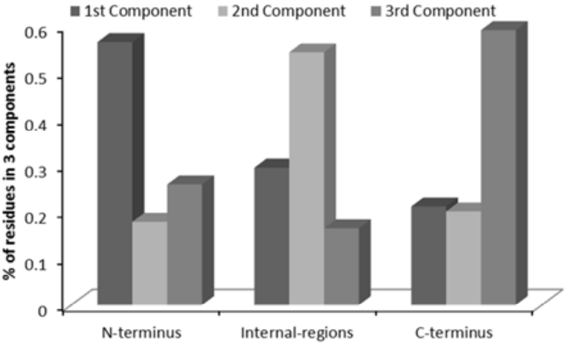


Fig. 3. Distribution of residues at N-terminus, internal-regions and at C-terminus with respect to three principal components identified by GMM-MAPML

5 Conclusion

In this paper we proposed a data-dependent probabilistic model for classification when given labels obtained by multiple noisy annotators but without any gold standard annotation. The proposed GMM-MAPML method uses a Gaussian mixture model (GMM) and Bayesian information criterion (BIC) to find the fittest model to approximate the distribution of the instances. Then the maximum a posterior (MAP) estimation of the hidden true labels and the maximum-likelihood (ML) estimation of quality of multiple annotators at each Gaussian component are provided alternately. Emotional speech classification and CASP9 protein disorder prediction experiments show a significant performance improvement of the proposed GMM-MAPML method as compared to the majority voting baseline and a previous data-independent method (MAP-ML). Moreover, GMM-MAPML also provides more accurate estimates of individual annotator performance for each Gaussian component, which can be used for active learning, feedback, and annotator selection.

The proposed method assumed that the annotators make their errors independently. We emphasize that in practice the independence assumption might not be always true which is the limitation of the proposed algorithm. To relax the independence assumption and to develop a more realistic model for the multiple-annotator problems, our research in progress includes additional parameters such as the degree of correlation among the annotators.

Acknowledgments. This work is funded in part by a grant with the Pennsylvania Department of Health. They specifically disclaim responsibility for any analyses, interpretations, or conclusions.

References

1. Amazon Mechanical Turk, `http://www.mturk.com`
2. Smyth, P., Fayyad, U.M., Burl, M.C., Perona, P., Baldi, P.: Inferring ground truth from subjective labelling of venus images. In: NIPS, pp. 1085–1092 (1994)
3. Jin, R., Ghahramani, Z.: Learning with multiple labels. In: NIPS, pp. 897–904 (2002)

4. Sheng, V.S., Provost, F.J., Ipeirotis, P.G.: Get another label? Improving data quality and data mining using multiple, noisy labelers. In: KDD, pp. 614–622 (2008)
5. Donmez, P., Carbonell, J.G.: Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In: CIKM, pp. 619–628 (2008)
6. Donmez, P., Carbonell, J.G., Schneider, J.: Efficiently learning the accuracy of labeling sources for selective sampling. In: KDD, pp. 259–268 (2009)
7. Crammer, K., Kearns, M., Wortman, J.: Learning from multiple sources. Journal of Machine Learning Research 9, 1757–1774 (2008)
8. Dekel, O., Shamir, O.: Vox populi: Collecting high-quality labels from a crowd. In: COLT (2009)
9. Snow, R., O'Connor, B., Jurafsky, D., Ng, A.Y.: Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. In: EMNLP, pp. 254–263 (2008)
10. Cholleti, S.R., Goldman, S.A., Blum, A., Politte, D.G., Don, S., Smith, K., Prior, F.: Veritas: combining expert opinions without labeled data. International Journal on Artificial Intelligence Tools 18, 633–651 (2009)
11. Raykar, V.C., Yu, S., Zhao, L.H., Jerebko, A.K., Florin, C., Valadez, G.H., Bogoni, L., Moy, L.: Supervised learning from multiple experts: whom to trust when everyone lies a bit. In: ICML, pp. 889–896 (2009)
12. Whitehill J., Ruvolo P., Wu T., Bergsma J., Movellan J.: Whose vote should count more: optimal integration of labels from labelers of unknown expertise. In NIPS (2009)
13. Welinder P., Branson S., Belongie S., Perona P.: The multidimensional wisdom of crowds. In: NIPS (2010)
14. Audhkhasi K., Narayanan S.: Data-dependent evaluator modeling and its application to emotional valence classification from speech. In: InterSpeech, pp. 2366–2369 (2010)
15. Rzhetsky, A., Shatkay, H., Wilbur, W.J.: How to get the most out of your curation effort. PLoS. Comput. Biol. 5(5), e1000391 (2009)
16. Zhang, P., Obradovic, Z.: Unsupervised integration of multiple protein disorder predictors. In: IEEE Int'l. Conf. Bioinformatics and Biomedicine, pp. 49–52 (2010)
17. Yan, Y., Rosales, R., Fung, G., Schmidt, M., Hermosillo, G., Bogoni, L., Moy, L., Dy, J.G.: Modeling annotator expertise: learning when everybody knows a bit of something. Journal of Machine Learning Research - Proceedings Track 9, 932–939 (2010)
18. Banfield, J.D., Raftery, A.E.: Model-based Gaussian and non-Gaussian clustering. Biometrics 49, 803–821 (1993)
19. Martinez, W.L., Martinez, A.R.: Exploratory data analysis with MATLAB, pp. 163–195. Chapman & Hall/CRC, Boca Raton (2004)
20. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B 39(1), 1–38 (1977)
21. Fraley, C., Raftery, A.E.: How many clusters? Which clustering method? Answers via model-based cluster analysis. Comput. J., 578–588 (1998)
22. Bishop, C.: Pattern recognition and machine learning, pp. 203–213. Springer, New York (2006)
23. Lee, S., Yildirim, S., Kazemzadeh, A., Narayanan, S.: An articulatory study of emotional speech production. In: Eurospeech, pp. 497–500 (2005)
24. VOICEBOX, `http://www.ee.imperial.ac.uk/hp/staff/dmb/voicebox/voicebox.html`
25. CASP experiments, `http://predictioncenter.org/`
26. Noivirt-Brik, O., Prilusky, J., Sussman, J.L.: Assessment of disorder predictions in CASP8. Proteins 77(suppl. 9), 210–216 (2009)
27. Uversky, V.N., Dunker, A.K.: Understanding protein non-folding. Biochim. Biophys. Acta 1804, 1231–1264 (2010)

iDVS: An Interactive Multi-document Visual Summarization System

Yi Zhang, Dingding Wang, and Tao Li

School of Computer Science,
Florida International University,
Miami, FL 33199, USA
{yzhan004,dwang003,taoli}@cs.fiu.edu

Abstract. Multi-document summarization is a fundamental tool for understanding documents. Given a collection of documents, most of existing multi-document summarization methods automatically generate a static summary for all the users using unsupervised learning techniques such as sentence ranking and clustering. However, these methods almost exclude human from the summarization process. They do not allow for user interaction and do not consider users' feedback which delivers valuable information and can be used as the guidance for summarization. Another limitation is that the generated summaries are displayed in textual format without visual representation. To address the above limitations, in this paper, we develop iDVS, a visualization-enabled multi-document summarization system with users' interaction, to improve the summarization performance using users' feedback and to assist users in document understanding using visualization techniques. In particular, iDVS uses a new semi-supervised document summarization method to dynamically select sentences based on users' interaction. To this regard, iDVS tightly integrates semi-supervised learning with interactive visualization for document summarization. Comprehensive experiments on multi-document summarization using benchmark datasets demonstrate the effectiveness of iDVS, and a user study is conducted to evaluate the users' satisfaction.

Keywords: interactive multi-document summarization, visualization.

1 Introduction

Multi-document summarization aims to generate a compressed summary by extracting information from a collection of documents sharing the same or similar topics. Since the Internet faces the data overload threat with the explosive increase of the documents, automatic multi-document summarization has attracted much attention and various summarization applications have emerged in recent years.

Current research on multi-document summarization often treats it as an unsupervised learning problem. Existing summarization methods usually involve sentence clustering and ranking to extract the most important sentences from the document collection to form the summaries automatically. Although these methods can efficiently generate short summaries of the documents, they have two major limitations:

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 569–584, 2011.

- **Lack of User Participation:** Summarization is a subjective task, i.e., different users may have different understanding and opinions on the same set of documents. However, most existing summarization methods exclude human from the summarization process, which is efficient in terms of reducing users' workload, but it is not desired since the generated summaries is user-independent, contradicting to the subjective nature of summarization. These methods do not allow for user interaction and do not consider users' feedback which delivers valuable information and can be used as the guidance for summarization.
- **Homogeneous Result Presentation:** Current document summarization systems present their results in textual format. As the size of the document collection increases, it becomes time-consuming, cumbersome, and labor-intensive for users to quickly locate important information or glean insights from the summary.

To address the above limitations, we develop iDVS, a visualization-enabled multi-document summarization system with users' interaction, to improve the summarization performance using users' feedback and to assist users in document understanding using visualization techniques. As an interactive document visual summarization system, iDVS lays out the sentence relationships in the documents visually, and iteratively selects sentences to create a summary incorporating users' feedback. In particular, an energy-based layout algorithm is used for sentence visualization, and a new semi-supervised document summarization method is proposed to dynamically select sentences based on users' interaction.

Note that visualization plays a vital role in iDVS. First, visualization facilitates user interaction. In iDVS, visualization techniques are used to display the sentence relationships and to help users locate critical information in a large text collection. Second, visual presentation and navigation of summary results can assist users in understanding large documents efficiently. Unlike existing work in visual text analysis, which focuses either on developing new text analytic algorithms and/or novel visualization techniques, iDVS aims to tightly integrate semi-supervised learning (for dynamic sentence selection) with interactive visualization (for user exploration) for supporting effective document summarization. We conduct extensive experiments on the benchmark datasets to demonstrate the effectiveness of iDVS on multi-document summarization and also perform user studies to evaluate the users' satisfaction.

The rest of this paper is organized as follows. Section 2 discusses the related work on multi-document summarization, semi-supervised learning and document visualization techniques. Section 3 introduces the framework of iDVS. Methods used in iDVS including the layout algorithms, paragraph selection, and semi-supervised document summarization are described in Section 4. Section 5 illustrates an example summarization process of iDVS. Section 6 demonstrates and discusses the experimental results comprehensively. A user study is conducted in Section 7. Finally, Section 8 concludes.

2 Related Work

2.1 Multi-document Summarization

Various multi-document summarization methods have been studied recently. Our discussion here focuses on extractive methods. The most commonly used methods are centroid based, which usually rank sentences in the document collection according to their scores calculated by a set of predefined features, such as term frequency-inverse sentence frequency (TF-ISF), sentence or term position, and number of keywords [37,29,23]. Another type of methods use sentence graph representation and select sentences based on the votes from their neighbors using ideas similar to PageRank [11]. Other methods include Latent semantic analysis (LSA) based summarization [13], Non-negative matrix factorization (NMF) based summarization [33], Conditional Random Field (CRF) based summarization [31], and hidden Markov model (HMM) based method [8]. Some query-based summarization systems are also proposed [12].

Most of the existing document summarization methods are unsupervised as described above. There are also a few studies applying supervised and semi-supervised methods to document summarization. Wong et al. [35] first extracted and combined various sentence features, and then applied supervised and semi-supervised classifiers to obtain the labels for all the sentences. Although some work considers users' opinions by analyzing user comments [19], few work has been reported on iteratively making use of users' feedback to improve the quality of the generated summaries. Most of current document summarization systems deliver results in textual format only, and there are limited attempts on document summarization to present the relationships among sentences visually.

2.2 Visual Text Analysis

The work on visual text analysis can be broadly divided into five different categories: 1) ***Meta-data visualization*** methods which focus on visualizing the meta data of text documents. For example, many techniques have been developed to visualize the relationships between the email senders and receivers in email corpora [28,26,21], and TileBars is used to visualize document length and query term frequency [16]. 2) ***Document visualization*** methods which focus on displaying document relationships. Typical systems for document visualization include the Galaxy of News [30], Jigsaw [32], and ThemeRiver [15]. 3) ***Word visualization*** methods which mainly show the text information at the text level. Typical systems for word visualization include TextArc (www.textarc.org), WordTree [34], and FeatureLens [10]. 4) ***Text Visual analytic*** methods which typically integrate visualization with some text analytic methods. For example, Ando et al. [3] developed a visualization-enabled multi-document summarization by iterative residual rescaling, Allan et al. [2] used the cluster visualization to help a user rapidly identify relevant documents, Liu et al. [25] presented an interactive, visual text analysis tool to support both top-down and bottom-up text analysis. 5) ***Visual Summarization*** methods which present a document summary using representative images. Google has released the "Image from the page" feature in its web search system using the images in a page to summarize the page. Jiao et al. [20] propose a visual summarization system using external images (which are not the images in the page) for search and re-finding tasks.

Our iDVS can be viewed as a text visual analytic method by integrating summarization and visualization and its focus is on generating better summary results with user interaction and feedbacks. Compared with existing text visualization systems, iDVS tightly combines semi-supervised learning techniques with interactive visualization to support user-centric document summarization. On one hand, iDVS uses visualizations to display summarization results and facilitates user interaction. On the other hand, it employs semi-supervised learning for users to perform dynamic sentence selection.

2.3 General Visualization Analytic

In general information visualization, researchers have developed a number of methods for visual analytics. For example, HD-Eye [18] and n23Tool [36] integrate visualization and clustering algorithms by representing the possible clusters and the relationship between clusters in each projection of any interactive system to visualize and analyze clusters. A visual framework VISTA is developed in [6] to visualize multi-dimensional datasets in a 2D star-coordinate space and to validate and refine the cluster structure. Chen and Liu [7] extended VISTA to an interactive visualization-based framework for clustering large datasets. More visualization techniques can be found in [14]. In our work, we focus on visualizing text data especially summary results. In addition, iDVS provides users with visual interactions tools to dynamic select sentences from multiple perspective.

2.4 Semi-Supervised Learning

In this paper, we treat document summarization given users' guidance as a problem of semi-supervised learning. Semi-supervised learning [5] learns from a set of partially labeled data points, including both labeled and unlabeled data, to predict the labels of the unlabeled data. For the document summarization task, we ask users to read and label a small portion of the documents, thus a semi-supervised algorithm can be developed to proceed document summarization with only a small portion of prior knowledge.

3 System Framework

Figure 1 shows the framework of iDVS. The system creates short summaries by selecting sentences iteratively based on the results of the semi-supervised document summarization engine and the feedback provided by users. Each iteration of the sentence selection includes the following steps.

- (1) The system generates a 2-D view graph of current sentence set of the given documents, in which each node represents a sentence, and the location and color of the sentence are determined by the layout and clustering algorithms respectively. Here, the initial "current sentence set" contains all the sentences in the documents.
- (2) Based on the layout results, the system picks out the most important sentence and asks a user to read the paragraph where the sentence locates.
- (3) The user partially orders the sentences contained in the paragraph based on their understanding of the content in the paragraph.
- (4) A semi-supervised sentence ranking algorithm by making use of the sentence preferences from the user is then performed to rank all the sentences in the documents. Top n sentences are then selected as candidates and recommended to the user. The number n is determined by the difference between the required summary length and the length of sentences already in the summary.

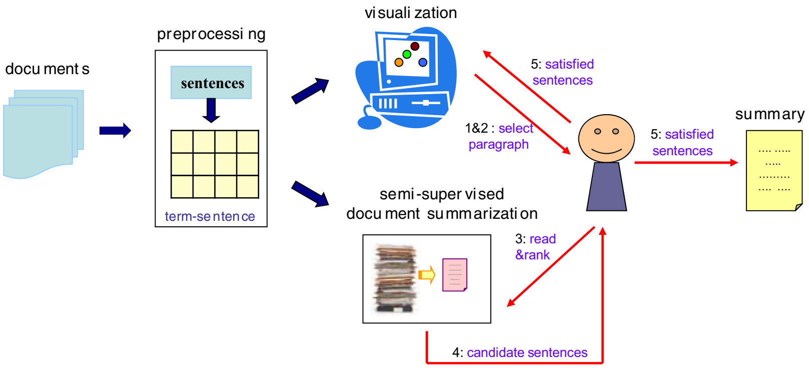


Fig. 1. The system overview

- (5) The user selects the candidate sentences which he/she is satisfied with, then these sentences are included in the summary. The selected sentences and the clusters that they belong to are then removed from current sentence set.
- (6) A new iteration starts until the required length of the summary is reached. One thing worth mentioning is that the number of sentence clusters and the number of sentences in a summary are small in general, so the involved amount of user efforts is acceptable in practice.

4 Methodology

In this section, we introduce the key techniques used in iDVS for sentence cluster layout, semi-supervised document summarization, and user interaction.

4.1 Sentence Cluster Layout

Sentence Graph Representation. Given a collection of documents, we first decompose them into sentences. An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is then constructed to represent the relationships among the sentences, where $\mathcal{V}$ is the vertex set and $\mathcal{E}$ is the edge set. Each vertex in $\mathcal{V}$ is a sentence, and each edge in $\mathcal{E}$ is associated with the cosine similarity between two sentences (vertices). Two vertices are connected if their cosine similarity is greater than 0.

Linlog Layout Algorithm. Here, we use Linlog, a popular energy-based layout algorithm[27], to display the sentence relationships and present the clusters in the separated shapes. The energy function in Linlog is

$$E(p) = \sum_{\{u,v\}:u \neq v} (\omega_{\{u,v\}} \|p_u - p_v\| - d_v d_u ln \|p_u - p_v\|),$$

where $\omega_{\{u,v\}}$ is the weight of the edge connecting vertices u and v, and d_u and d_v are the degrees of u and v respectively. The optimal positions p of all the vertices are obtained by minimizing $\mathcal{E}$.

Clustering with Maximum Modularity. The positions of nodes (sentences) displayed by the energy-based layout algorithm are consistent with the clustering results obtained by maximizing graph modularity.

Modularity [27,1] can be defined as

$$\sum_{c \in C \in} [\frac{w_c}{w_C} - (\frac{d(c)^2}{d(C)^2})],$$

where w_c, w_C are the sum of edge weights in cluster c and cluster set C respectively, and $d(c)$ and $d(C)$ are the sum of node degrees for all the nodes in cluster c and cluster set C.

The clustering results can be easily obtained by a bottom-up algorithm, in which each sentence is treated as a singleton cluster at the outset and then successively merge pairs of clusters until the maximum modularity is reached.

4.2 Paragraph Selection and Users' Annotation

The visualization in iDVS clearly illustrates the following information for users. (1) The radius of each node is determined by the sentence's degree. The larger the node, the more important the corresponding sentence. In other words, the largest node in a cluster corresponds to the most important sentence in that cluster. (2) Large nodes in the overlapping area of two clusters may be the transition sentences between the clusters. (3) The larger the distance between two clusters, the dissimilar the two topics. Based on the visualization of sentence relationships, the user finds the largest nodes in each cluster, then picks any one of them, and the system returns its corresponding sentence with the paragraph that the sentence belongs to for the user to read and annotate.

In the annotation, the user is asked to read the selected paragraph and provide the comparable sentence pairs in the paragraph and their preferences of the two sentences based on their understanding of the content in the paragraph. The user finally provides all the comparable sentence pairs and label their preference of the two sentences for each sentence pair in the paragraph. Here the "comparable sentence pair" means that two sentences share similar topics. Thus, we have a number of sentence pairs and their preference, e.g.(s_i, s_j) and their relationship $r_i \geq r_j$ representing sentence i is preferred to be included in the summary comparing to sentence j. We treat the sentence pair (s_i, s_j) with their preference $r_i \geq r_j$ as a "sentence preference pair".

4.3 Semi-Supervised Document Summarization

For the summarization task, we are given a set of sentences $\mathcal{S} = \{s_i\}_{i=1}^m$ associated with their importance $\{r_i\}_{i=1}^m$. Through users' interactions, we know some pairwise relationships

$$\mathcal{C} = \{(s_{k_i}, s_{k_j})\}_{k=1}^K,$$

such that if $(s_{k_i}, s_{k_j}) \in \mathcal{C}$, then $r_{k_i} \geq r_{k_j}$, where k is the index of the constraint and K is the total number of constraints. To apply the semi-supervised summarization method, we should first have a sentence graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$constructed as in Section 4.1. Then our semi-supervised model assumes that the estimated sentences preferences $\{r_i\}$ should vary sufficiently smooth with respect to $\mathcal{G}$. Usually such smoothness can be calculated by

$$\delta = \sum_{s_i \sim s_j} w_{ij} \left(\frac{r_i}{\sqrt{d_i}} - \frac{r_j}{\sqrt{d_j}} \right)^2 . \tag{1}$$

Here $d_i = \sum_j w_{ij}$ is the degree of sentence s_i, and $s_i \sim s_j$ denotes that s_i and s_j are adjacent (i.e., there is an edge linking them). Written in its matrix form, Eq.(1) can be transformed to

$$\delta = \mathbf{r}^T \mathbf{L} \mathbf{r}, \tag{2}$$

where

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}, \tag{3}$$

is the normalized Laplacian matrix [38], and $\mathbf{I}$ is an $m \times m$ identity matrix. Some theoretical research [4][17] showed that if we assume the sentences in $\mathcal{S}$ are sampled from some continuous "sentence manifold" $\mathcal{M}_{\mathcal{S}}$, then when the number of sampled sentences tends to infinity, the normalized Laplacian will converge to the Laplace-Beltrami operator on $\mathcal{M}_{\mathcal{S}}$, which is strongly related to the functional smoothness defined on $\mathcal{M}_{\mathcal{S}}$.

In our semi-supervised summarization scenario, we assume that we have an "initial" preference vector $\mathbf{r}_0$ which is estimated via some unsupervised methods, and our method can be viewed as a way to "improve" such initial guess $\mathbf{r}_0$ by (1) the sentence-sentence geometric structure; (2) prior knowledge on sentence preferences. For simplicity, we use Euclidean distance to measure the discordance of the final preference vector $\mathbf{r}$ and initial $\mathbf{r}_0$ whose objective is to solve the following optimization problem

$$\min_{\mathbf{r}} \|\mathbf{r} - \mathbf{r}_0\|^2 + \lambda \mathbf{r}^T \mathbf{L} \mathbf{r}, r_{k_i} \geq r_{k_j}, \ \forall \ (s_{k_i}, s_{k_j}) \in \mathcal{C}, \tag{4}$$

where $\mathbf{r} = [r_1, r_2, \cdots, r_m]^T$ and has been normalized to 1, and $\lambda > 0$ is the regularization parameter to tradeoff the loss and smoothness terms. If we denote $\mathbf{e}_{k_i} \in \mathbf{R}^{m \times 1}$ as an all-zero vector with only its k_i-th element being 1, then we can rewrite problem (4) in its standard form as

$$\min_{\mathbf{r}} \|\mathbf{r} - \mathbf{r}_0\|^2 + \lambda \mathbf{r}^T \mathbf{L} \mathbf{r}, \mathbf{r}^T(\mathbf{e}_{k_i} - \mathbf{e}_{k_j}) \geq 0, \ \forall \ (s_{k_i}, s_{k_j}) \in \mathcal{C}. \tag{5}$$

Clearly, the objective of the above problem is quadratic in $\mathbf{r}$ and the constraints are linear in $\mathbf{r}$, which makes it a convex quadratic programming problem whose global optimum can be easily found by some mature numerical methods.

The objective of problem (5) can also be understood from a probabilistic perspective, where we have the initial guess $\mathbf{r}_0$, the sentences $\mathcal{S}$, and we want to learn the true sentence preferences $\mathbf{r}$. In general, the optimal $\mathbf{r}$ can be obtained as the one with the maximum posterior probability

$$P(\mathbf{r}|\mathcal{S}, \mathbf{r}_0) = \frac{1}{Z} P(\mathbf{r}|\mathcal{S}) P(\mathbf{r}|\mathbf{r}_0),$$

where

$$P(\mathbf{r}|\mathcal{S}) = \frac{1}{Z_1}\exp(-\mathbf{r}^T\mathbf{L}\mathbf{r}),$$

is the prior probability of $\mathbf{r}$ given the sentence set $\mathcal{S}$,

$$P(\mathbf{r}|\mathbf{r}_0) = \frac{1}{Z_2}\exp(-\alpha\|\mathbf{r}-\mathbf{r}_0\|^2)$$

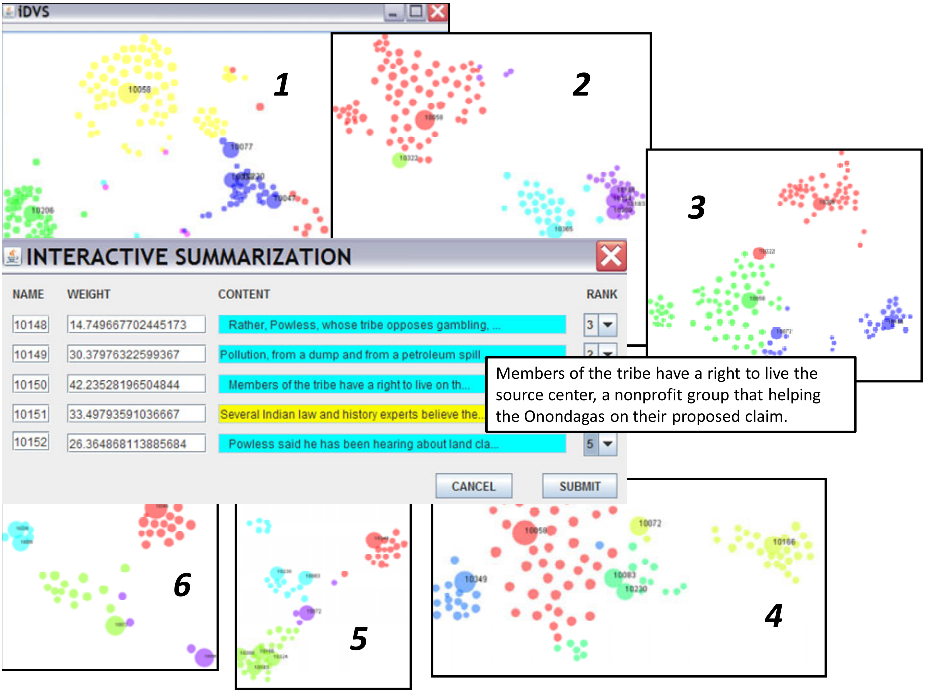
is the likelihood of $\mathbf{r}$ given the initial guess $\mathbf{r}_0$, and Z_1, Z_2, Z, α are constants. If we define the energy E as

$$E = \mathbf{r}^T\mathbf{L}\mathbf{r} + \alpha\|\mathbf{r}-\mathbf{r}_0\|^2.$$

Then maximizing the posterior $P(\mathbf{r}|\mathcal{X}, \mathbf{r}_0)$ is equivalent to minimizing the energy E. Let $\lambda = \frac{1}{\alpha}$, we can get the objective in problem (5). Then we sort $\mathbf{r}$ in descending order to get the ranking of the sentences in the document collection, and the top sentences are selected as the candidate sentences and returned to the user.

5 An Illustrative Example

Figure 2 illustrates an example of the visualization results in each iteration of the summary generation process. Some key features of iDVS can be demonstrated in this example as follows.



INTERACTIVE SUMMARIZATION

NAME	WEIGHT	CONTENT	RANK
10148	14.749667702445173	Rather, Powless, whose tribe opposes gambling, ...	3
10149	30.37976322599367	Pollution, from a dump and from a petroleum spill	2
10150	42.23528196504844	Members of the tribe have a right to live on th...	
10151	33.49793591036667	Several Indian law and history experts believe the...	
10152	26.364868113885684	Powless said he has been hearing about land cla...	5

Members of the tribe have a right to live the source center, a nonprofit group that helping the Onondagas on their proposed claim.

Fig. 2. An illustrative example

- Given a set of 25 news articles, the system demonstrates the sentence clusters using the layout and clustering algorithms described in Section 4.1. The layout results of the original sentence set are shown in the upper left part of Figure 2. The most important sentences (nodes) in each cluster are of large size and labeled with their sentence IDs. In this example, there exist six clusters in the original sentence set, each of which is associated with a unique color.
- As the sizes of nodes and clusters indicate the importance of a sentence and a sentence cluster, users select one node from the graph, the system returns the corresponding sentence and the paragraph it belongs to, each sentence in the graph is displayed in a panel as shown in the lower right part in Figure 2. Note that sentences in the same paragraph are not necessary to be in the same cluster, thus different colors in the panel indicate the sentence clusters. For the space limit, only partial of the sentences are displayed, and if users are interested in any sentences, they can move the mouse over the sentence, and a tip will pop up to show the entire sentence. Users can read and rank the sentences and then submit their preferences through the system to the semi-supervised document summarization engine.
- In each iteration, the summarization engine will generate candidate sentences using the proposed semi-supervised document summarization method (as described in Section 4.3). Users can pick their interested sentences to include in the summary.
- Then the system will remove those sentences and their clusters which have been added into the summary, and a new iteration starts. Note that if the user is not satisfied with any candidate sentence, no nodes will be removed and the layout won't change. However, the user can either re-rank the sentences in the most important paragraph or the system can select the second important paragraph for users to annotate. In this example, there are three iterations, and from the figure, we clear observe the reduction of number of sentences in each iteration.

6 Experiments

6.1 Data Description

To evaluate the summarization results empirically, we use the DUC2002 and DUC2004 data sets, both of which are open benchmark data sets from Document Understanding Conference (DUC) for generic automatic summarization evaluation. Table 1 gives a brief description of the data sets.

Table 1. Description of the data sets for multi-document summarization

	DUC2002	DUC2004
number of document collections	59	50
number of documents in each collection	∼10	10
data source	TREC	TDT
summary length	200 words	665bytes

6.2 Implemented Summarization Systems

We implement the following unsupervised and supervised document summarization methods as the baseline systems to examine the quality of the summaries generated by our proposed method: 1) **Random**: selects sentences randomly for each document collection. (2) **Centroid**: applies MEAD algorithm proposed in [29] using centroid value, positional value, and first-sentence overlap as features. (3) **LexPageRank**: a graph-based summarization method recommending sentences by the voting of their neighbors [11]. (4) **LSA**: conducts latent semantic analysis on terms by sentences matrix as proposed in [13]. (5) **NMF**: performs NMF on terms by sentences matrix and ranks the sentences by their weighted scores [22]. (6) **KM**: performs K-means algorithm to clustering the sentences and chooses the center sentences in each clusters. (7) **Supervised**: transfers the document summarization problem to a two-class sentence classification problem (i.e. one class represents the sentences included in the summary, while the other class represents the rest of the sentences). Then Naive Bayesian classifier (NBC) is used as the learning approach. The class label information comes from one human summary in DUC for evaluation [35]. (8) **iDVS**: our proposed semi-supervised document summarization method as described in Section 4.

Table 2. Overall performance comparison on DUC2002 data using ROUGE evaluation methods

Systems	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-W	ROUGE-SU
DUC Best	**0.49869**	**0.25229**	**0.46803**	**0.20071**	**0.28406**
Random	0.38475	0.11692	0.37218	0.15941	0.18057
Centroid	0.45379	0.19181	0.43237	0.17971	0.23629
LexPageRank	0.47963	0.22949	0.44332	0.18978	0.26198
LSA	0.43078	0.15022	0.40507	0.15220	0.20226
NMF	0.44587	0.16280	0.41513	0.16072	0.21687
KM	0.43156	0.15135	0.40376	0.15038	0.20144
Consistency Method	0.46449	0.22173	0.44514	0.18311	0.25670
Variant Consistency 1	0.49990	0.26461	0.47233	0.20118	0.28745
Variant Consistency 2	0.47121	0.23322	0.42578	0.17266	0.23007
Homonic-CMN	0.46809	0.23039	0.41759	0.17556	0.22876
Green's Function	0.46591	0.22788	0.41113	0.16092	0.22227
Supervised	**0.51243**	**0.26538**	**0.47731**	**0.20827**	**0.29634**
iDVS	**0.50612**	**0.25981**	**0.47535**	**0.20279**	**0.28863**

Five Semi-supervised Summarization Systems: In addition to the above systems, we also compare our *interactive* semi-supervised document summarization system (iDVS) with the following baseline summarization systems which directly make use of semi-supervised learning methods. In these baseline systems, the most important sentence from each sentence cluster obtained using clustering with maximum modularity (as described in Section 4.2) is labeled as summary sentence. Then semi-supervised learning approaches are used to generate a cluster C with size n_C whose elements are the candidate summary sentences. Finally, a sentence subset K with size n_K is selected from C to form the final summary using Eq.(6):

$$\min \frac{\gamma}{n_C \times n_K} \sum_{s_i \in K} [\sum_{s_j \in C} Sim(s_i, s_j)] - \frac{1-\gamma}{n_K \times n_K} \sum_{s_i, s_j \in K} Sim(s_i, s_j), \quad (6)$$

where the first term is the average similarity between the selected sentence and the rest sentences and the second term is the average pairwise sentence similarity in K, both are calculated using cosine similarity. γ is a weight parameter, and is set to 0.6 empirically. In other words, the final sentence selection aims to choose sentences that have high similarity values with all other sentences in C and low similarity values with the selected sentences. Five popular semi-supervised learning approaches are used in the baseline systems: (1-3) the algorithm proposed in [35] which conducts semi-supervised learning with local and global consistency (**Consistency Method**), and two of its variants (**Variant Consistency 1** and **Variant Consistency 2**); (4) Harmonic Gaussian field method coupled with the Class Mass Normalization (**Harmonic-CMN**) [39]; (5) Green's function learning algorithm (**Green's Function**) Proposed in [9].

6.3 Evaluation Method

We use ROUGE [24] toolkit (version 1.5.5) to measure the summarization performance, which is widely applied by DUC for performance evaluation. It measures the quality of a summary by counting the unit overlaps between the candidate summary and a set of reference summaries. Intuitively, the higher the scores, the more similar between the two summaries. As we have similar conclusions for different scores, for simplicity, in this paper, we only report the average F-measure scores generated by ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-W and ROUGE-SU to compare the implemented systems. One thing worth mentioning is that since different users might have different perceptions of which sentence is more important, the results of iDVS reported in the experiments are the average scores obtained from three users.

6.4 Experimental Results

Overall Performance. First of all, we compare the overall performance of our proposed semi-supervised document summarization method with other most widely used unsupervised and supervised summarization methods. Table 2 and Table 3 show the ROUGE evaluation results on DUC2002 and DUC2004 data sets respectively. We set λ to be 0.6 empirically. From the results, we have the following observations: (1) Our iDVS method outperforms all the unsupervised summarization methods. This benefits from the guidance of the preference and feedback provided by users and our proposed semi-supervised learning method which makes use of the user interaction. (2) We compare our method with the best team from DUC competition. Although the advanced natural language processing techniques are used by the best team, our semi-supervised method still outperforms it, which demonstrates the effectiveness of our method utilizing a small amount of labeled data. (3) Compared with the baseline summarization systems which directly make use of semi-supervised learning, our iDVS method is able to utilize user interactions and achieves the best performance. The performance improvement clearly demonstrate the advantage of user participation in our system. (4) Our method maintains the comparable performance with the supervised summarization approach which makes use of the human generated summaries. Although the results from supervised learning method are encouraging, a fully labeled training data is too expensive and usually infeasible to obtain.

To better demonstrate the results, Figure 3 visually illustrate the comparison. As we have similar conclusion on different ROUGE scores, due to the space limit, we only show the ROUGE-1 results in these figures.

Table 3. Overall performance comparison on DUC2004 data using ROUGE evaluation methods

Systems	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-W	ROUGE-SU
DUC Best	**0.38224**	**0.09216**	**0.38687**	**0.13325**	**0.13233**
Random	0.31865	0.06377	0.34521	0.11734	0.11779
Centroid	0.36728	0.07379	0.36182	0.12439	0.12511
LexPageRank	0.37842	0.08572	0.37531	0.13121	0.13097
LSA	0.34145	0.06538	0.34973	0.12042	0.11946
NMF	0.36747	0.07261	0.36749	0.12961	0.12918
KM	0.34872	0.06937	0.35882	0.12339	0.12115
Consistency Method	0.37565	0.09776	0.37781	0.12765	0.13000
Variant Consistency 1	0.39225	0.10477	0.39547	0.13762	0.13651
Variant Consistency 2	0.37740	0.09803	0.38006	0.12812	0.13091
Homonic-CMN	0.37211	0.09466	0.37094	0.12233	0.12997
Green's Function	0.36411	0.07215	0.36645	0.12470	0.12261
Supervised	**0.39622**	**0.10973**	**0.39956**	**0.14221**	**0.14095**
iDVS	**0.39227**	**0.10524**	**0.39633**	**0.13816**	**0.13751**

Paragraph Selection. In the experiments, we select paragraphs based on users' feedback and the visualized structure of sentences. In this set of experiments, we examine this paragraph selection method and compare it with the following selection solutions: (a) **RandomSel**: randomly selects paragraphs of the documents; and (b) **IdealSel**: selects paragraphs containing the sentences in the human summaries used for evaluation. Figure 4 demonstrates the comparison results. From the results, we can see that our paragraph selection method is more effective than the random selection. Although the ideal selection performs the best, in real application the human summaries are infeasible to obtain, while our selection method is simple and cost efficient.

Parameter Tuning. In Figure 5, we gradually tune the parameter λ in our method to adjust the weights between the two parts of the objective function in 5. We vary λ from 0.1 to 0.9 in every 0.1 interval. From the results, we can see that when λ is 0.6, the generated summaries achieve the highest performance.

7 A User Survey

To better evaluate the results of iDVS, we conduct a user survey. The subjects of the survey are fifteen students at different levels and from various majors of a university.

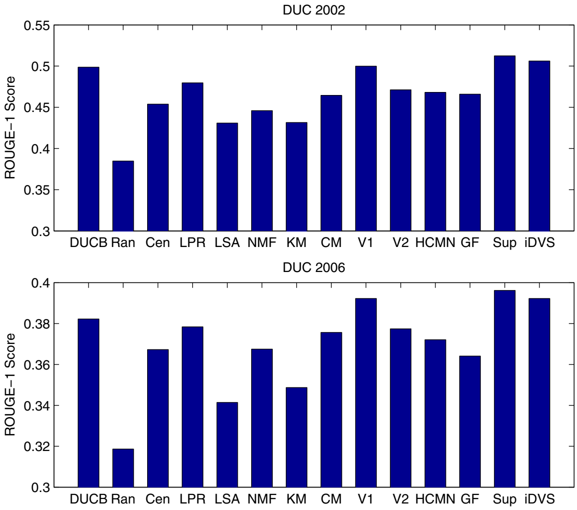


Fig. 3. Overall summarization performance on DUC2002 data (Top), DUC2006 data (Bottom) using ROUGE-1. DUCB, Ran, Cen, LPR, LSA, NMF, CM, V1, V2, HCMN, GF, Sup are corresponding to the summarization methods named DUC Best, Random, Centroid, LexPageRank, LSA, NMF, Consistency Method, Variant Consistency 1, Variant Consistency 2, Homonic-CMN, Green's Function and Supervised respectively.

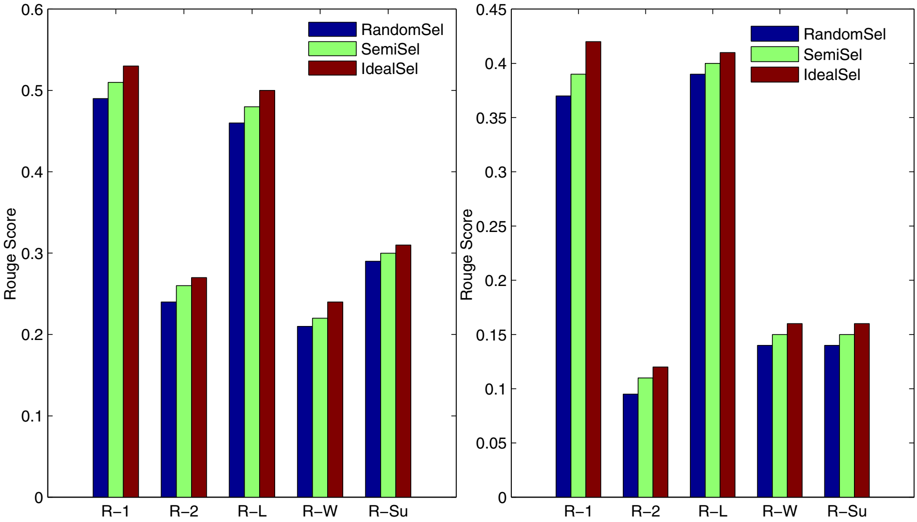


Fig. 4. iDVS results with different paragraph selection methods on DUC2002 data (left) and DUC2004 data (right)

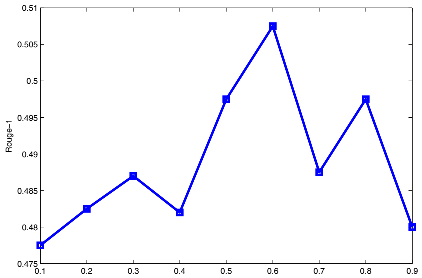


Fig. 5. iDVS parameter tuning using DUC2002 data

Table 4. User satisfaction comparison

Systems	Scores
iDVS	4.07
Random	1.47
Centroid	2.53
LexPageRank	2.80
Supervised	3.47

Each participant randomly selects a set of news documents, and use iDVS to form a summary. Then they are asked to assign a score of 1 (least satisfaction) to 5 (highest satisfaction), according to their satisfaction of the use of iDVS. We compare iDVS with some typical summarization systems including Random, Centroid, LexPageRank, and Supervised. The first three methods are unsupervised methods, and the last one is supervised method. And all of these systems do not support visualization of the sentence relationships. Table 4 demonstrate the satisfaction scores for each system, and the results show the effectiveness and high usability of iDVS. From the results, we have the following observations: (1) Generally, users' satisfaction scores are consistent with the performance of the summarization systems. (2) Users are highly satisfied with iDVS because (1) the generated summaries are based on the guidance and feedback through users' interaction, i.e., it supports personalized opinions; (2) iDVS provides visualization to illustrate the latent relationships among sentences.

8 Conclusion

In this paper, we develop iDVS, an interactive document visual summarization system, to visualize the structure of sentences contained in a document collection, and generate a short summary based on users' opinions iteratively. The main contributions of iDVS are that (1) the visual system can help users better understand the latent relationships in the documents/sentences easily; (2) the summarization process incorporates users' feedback including some domain knowledge and personalized opinions, which can validate and refine the summarization results effectively. Comprehensive experiments and a user study demonstrate the effectiveness of iDVS.

Acknowledgement. The work is partially supported by US National Science Foundation under grants IIS-0546280 and CCF-0830659.

References

1. Agarwal, G., Kempe, D.: Modularity-maximizing graph communities via mathematical programming. The European Physical Journal B - Condensed Matter and Complex Systems 66(3), 409–418 (2008)
2. Allan, J., Leouski, A.V., Swan, R.C.: Interactive cluster visualization for information retrieval. In: ECDL (1998)
3. Ando, R., Boguraev, B., Byrd, R., Neff, M.: Visualization-enabled multi-document summarization by iterative residual rescaling. Nat. Lang. Eng. 11(1), 67–86 (2005)
4. Belkin, M., Niyogi, P.: Towards a theoretical foundation for laplacian-based manifold methods. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 486–500. Springer, Heidelberg (2005)
5. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press, Cambridge (2006)
6. Chen, K., Liu, L.: Vista: validating and refining clusters via visualization. Information Visualization 3(4), 257–270 (2004)
7. Chen, K., Liu, L.: ivibrate: Interactive visualization-based framework for clustering large datasets. ACM Trans. Inf. Syst. 24(2), 245–294 (2006)
8. Conroy, J., O'Leary, D.: Text summarization via hidden markov models. In: SIGIR, pp. 406–407 (2001)
9. Ding, C., Jin, R., Li, T., Simon, H.D.: A learning framework using green's function and kernel regularization with application to recommender system. In: SIGKDD (2007)
10. Don, A., Zheleva, E., Gregory, M., Tarkan, S., Auvil, L., Clement, T., Shneiderman, B., Plaisant, C.: Discovering interesting usage patterns in text collections: integrating text mining with visualization. In: CIKM, pp. 213–222 (2007)
11. Erkan, G., Radev, D.: Lexpagerank: Prestige in multi-document text summarization. In: EMNLP (2004)
12. Goldstein, J., Kantrowitz, M., Mittal, V., Carbonell, J.: Summarizing text documents: Sentence selection and evaluation metrics. In: SIGIR, pp. 121–128 (1999)
13. Gong, Y., Liu, X.: Generic text summarization using relevance measure and latent semantic analysis. In: SIGIR, pp. 75–95 (2001)
14. Grinstain, G., Ankerst, M., Keim, D.: Visual data mining: Background, applications, ad drug discovery applications. In: SIGMOD (1999)
15. Havre, S., Hetzler, E., Whitney, P., Nowell, L.: Themeriver: Visualizing thematic changes in large document collections. IEEE Transactions on Visualization and Computer Graphics 8(1), 9–20 (2002)
16. Hearst, M.A.: Tilebars: visualization of term distribution information in full text information access. In: CHI, pp. 59–66 (1995)
17. Hein, M., Audibert, J., Von Luxburg, U.: From graphs to manifolds - weak and strong pointwise consistency of graph laplacians. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 470–485. Springer, Heidelberg (2005)
18. Hinneburg, A., Keim, D., Wawryniuk, M.: Visual mining of high-dimensional data. IEEE Computer Graphics and Applications (1999)
19. Hu, M., Sun, A., Lim, E.-P.: Comments-oriented document summarization: understanding documents with readers' feedback. In: SIGIR, pp. 291–298 (2008)

20. Jiao, B., Yang, L., Xu, J., Wu, F.: Visual summarization of web pages. In: SIGIR, pp. 499–506 (2010)
21. Kerr, B.: Thread arcs: an email thread visualization. In: InfoVis, pp. 211–218 (2003)
22. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS (2001)
23. Lin, C.-Y., Hovy, E.: From single to multi-document summarization: A prototype system and its evaluation. In: ACL, pp. 457–464 (2001)
24. Lin, C.-Y., Hovy, E.: Automatic evaluation of summaries using n-gram co-occurrence statistics. In: NLT-NAACL, pp. 71–78 (2003)
25. Liu, S., Zhou, M.X., Pan, S., Qian, W., Cai, W., Lian, X.: Interactive, topic-based visual text summarization and analysis. In: CIKM, pp. 543–552 (2009)
26. Nardi, B.A., Whittaker, S., Isaacs, E., Creech, M., Johnson, J., Hainsworth, J.: Integrating communication and information through contactmap. Commun. ACM 45(4), 89–95 (2002)
27. Noack, A.: Modularity clustering is force-direced layout. Physical Review E 79, 026102 (2009)
28. Perer, A., Smith, M.A.: Contrasting portraits of email practices: visual approaches to reflection and analysis. In: AVI 2006, pp. 389–395 (2006)
29. Radev, D., Jing, H., Stys, M., Tam, D.: Centroid-based summarization of multiple documents. In: Information Processing and Management, pp. 919–938 (2004)
30. Rennison, E.: Galaxy of news: an approach to visualizing and understanding expansive news landscapes. In: UIST 1994, pp. 3–12 (1994)
31. Shen, D., Sun, J.-T., Li, H., Yang, Q., Chen, Z.: Document summarization using conditional random fields. In: IJCAI, pp. 2862–2867 (2007)
32. Stasko, J., Görg, C., Liu, Z.: Jigsaw: supporting investigative analysis through interactive visualization. Information Visualization 7(2), 118–132 (2008)
33. Wang, D., Li, T., Zhu, S., Ding, C.H.Q.: Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In: SIGIR, pp. 307–314 (2008)
34. Wattenberg, M., Viégas, F.B.: The word tree, an interactive visual concordance. IEEE Transactions on Visualization and Computer Graphics 14(6), 1221–1228 (2008)
35. Wong, K.-F., Wu, M., Li, W.: Extractive summarization using supervised and semi-supervised learning. In: Coling (2008)
36. Yang, L.: n23tool: A tool for exploring large relational datasets through 3d dynamic projections. In: CIKM (2000)
37. Yih, W.-T., Goodman, J., Vanderwende, L., Suzuki, H.: Multi-document summarization by maximizing informative content-words. In: IJCAI, pp. 1776–1782 (2007)
38. Zhou, D., Bousquet, O., Navin Lal, T., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: NIPS, vol. 16, pp. 321–328 (2004)
39. Zhu, X.: Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison (2005)

Discriminative Experimental Design

Yu Zhang and Dit-Yan Yeung

Hong Kong University of Science and Technology
{zhangyu,dyyeung}@cse.ust.hk

Abstract. Since labeling data is often both laborious and costly, the labeled data available in many applications is rather limited. Active learning is a learning approach which actively selects unlabeled data points to label as a way to alleviate the labeled data deficiency problem. In this paper, we extend a previous active learning method called transductive experimental design (TED) by proposing a new unlabeled data selection criterion. Our method, called discriminative experimental design (DED), incorporates both margin-based discriminative information and data distribution information and hence it can be seen as a discriminative extension of TED. We report experiments conducted on some benchmark data sets to demonstrate the effectiveness of DED.

1 Introduction

It is not uncommon that the labeled data available in many machine learning applications is rather limited because the labeling process is both laborious and costly. We refer to this as the labeled data deficiency problem. However, even though labeled data is scarce, abundant unlabeled data may be available in some applications at very low cost. There exist some learning approaches which exploit unlabeled data to boost the generalization performance. One of them is semi-supervised learning [1] which exploits information contained in the unlabeled data such as the geometric structure of the data. Another approach is active learning [2,3] which expands the labeled data set while keeping the labeling cost low by selecting only the most representative unlabeled data points to label.

Unlike many conventional machine learning methods which wait passively for labeled data to be provided in order to start the learning process, active learning takes a more active approach by selecting unlabeled data points to query some oracle or domain expert. As a result, the expanded labeled data set can help the system learn a better, more accurate model. The typical learning procedure of the active learning approach is depicted in Table 1. Most existing active learning methods, such as support vector machine (SVM) active learning [4,5,6], select only one data point in each active learning iteration, i.e., the set $\mathcal{S}$ in Table 1 is a singleton set. To select multiple data points, multiple iterations are needed and hence the learning model has to be re-trained multiple times, incurring high computational cost. In recent years, a few active learning methods have been proposed to select multiple data points in each iteration to reduce the total computational cost. Some examples include batch mode active learning [7,8] and transductive experimental design (TED) [9,10].

TED has been demonstrated to be an effective method for active learning. However, it can only utilize information about the data distribution. We propose here an

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 585–596, 2011.

Table 1. Typical Active Learning Procedure

```
Input: Labeled data set $\mathcal{L}$; Unlabeled data set $\mathcal{U}$
Output: Learning model
Step 1: Train a learning model based on $\mathcal{L}$;
Step 2:
For $t = 1, \ldots, t_{\max}$
    2.1: Select an unlabeled data set $\mathcal{S}$ from $\mathcal{U}$ based
         on some unlabeled data selection criterion;
    2.2: Query an oracle to label $\mathcal{S}$;
    2.3: $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{S}, \mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}$;
    2.4: Re-train the learning model based on $\mathcal{L}$;
```

extension of TED, called discriminative experimental design (DED), which combines the strengths of both SVM active learning and TED. In particular, the data selection criterion of DED incorporates both margin-based discriminative information and data distribution information. Under the DED framework, we will show that TED can be seen as a special case by treating all data points as equally important. To solve the DED learning problem, we propose a new optimization procedure which exhibits some interesting properties. We will report experiments conducted on some benchmark data sets to demonstrate the effectiveness of DED.

In the next section, we will briefly review active learning and TED. We then present DED in section 3 as a discriminative extension of TED. Section 4 presents our experimental results and then the final section concludes the paper.

2 Active Learning and TED

Among the most important elements of active learning is the unlabeled data selection criterion which has attracted a lot of attention in the machine learning research community. The most commonly used selection criteria include uncertainty sampling [11], query-by-committee [12], representative sampling [13] and Bayesian error reduction [14]. Among these criteria, uncertainty sampling is the most widely studied one. A representative method that uses this criterion is SVM active learning which uses the decision function value as the uncertainty measure for guiding the selection of unlabeled data points. Although SVM active learning performs well in many applications, it does have some limitations. Since it only considers data points lying near the decision boundary of the current classifier, it ignores information about the whole data distribution but such information has been shown to be effective for active learning in representative sampling and TED. Moreover, since labeled data points are often scarce during the early stage of learning, estimation of the margin is not very accurate and hence SVM active learning may select atypical data points or even outliers. Furthermore, since SVM active learning selects only one data point in each iteration, the model has to be re-trained multiple times during the active learning procedure.

TED, which has its origin in experimental design [15] from the statistics community, is used for active learning in [9]. The learning procedure of TED is somewhat different from that of conventional active learning in that it does not assume the existence of labeled data before active learning begins and hence its data selection criterion does not rely on discriminative information provided by the current classifier. By utilizing the data distribution information, TED can choose multiple representative data points in each iteration of the active learning procedure.

It appears that both conventional active learning methods and TED have advantages that are complementary to each other. In the next section, we will propose a new method that combines the strengths of conventional active learning and TED. In particular, the new data selection criterion will incorporate both margin-based discriminative information and data distribution information.

3 Discriminative Experimental Design

Suppose we are given a training set $\mathcal{D}$ which contains both labeled and unlabeled data. The labeled part of $\mathcal{D}$ consists of l labeled data points $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, l$, where $\mathbf{x}_i \in \mathbb{R}^d$ and its corresponding class label $y_i \in \{-1, 1\}$. The unlabeled part of $\mathcal{D}$ consists of u unlabeled data points $\mathbf{x}_j \in \mathbb{R}^d$, $j = l+1, \ldots, l+u$. Usually $l \ll u$ because labeling data is laborious and costly. We assume that the data points are centered and the classification function is defined as $f(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\cdot)$ denotes the feature map corresponding to some kernel function $k(\cdot, \cdot)$. In general, $\boldsymbol{\phi}(\cdot)$ may have no explicit form but is only defined implicitly.

3.1 Objective Function

As discussed in the previous section, the goal of this paper is to exploit the advantages of both SVM active learning and TED in defining DED. This property should be reflected by the objective function of DED which is what we will look at in this subsection.

The learning setting of DED is similar to that of conventional active learning as depicted in Table 1, which has both labeled and unlabeled data before active learning begins. We hope to define a better data selection criterion which, on one hand, incorporates discriminative information from the labeled data and, on the other hand, incorporates data distribution information from the unlabeled data.

Let us first review the least squares SVM [16] which is used, though in different ways, by both TED and DED. The optimization problem can be stated as follows:

$$\min_{\mathbf{w}} \sum_{i=1}^{l} (\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|_2^2, \tag{1}$$

where $\lambda > 0$ is the regularization parameter and $\|\cdot\|_2$ denotes the 2-norm for vectors. Since $y_i \in \{-1, 1\}$, (1) is equivalent to the following problem:

$$\min_{\mathbf{w}} J(\mathbf{w}) = \sum_{i=1}^{l} (1 - y_i \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i))^2 + \lambda \|\mathbf{w}\|_2^2. \tag{2}$$

We use the objective function $J(\mathbf{w})$ in (2) to learn the model parameters $\mathbf{w}$ for the classifier. Here we use the squared loss $L(s, t) = (1 - st)^2$ which is similar to the squared hinge loss $L'(s,t) = \max(0, 1 - st)^2$ used for SVM [17]. Similar to the squared hinge loss, the squared loss used here enforces the prediction of the classifier and the ground truth to have the same sign and that there is a large margin between the positive and negative classes. Moreover, it is equivalent to the conventional squared loss $L(s,t) = (s-t)^2$ for binary classification problems and it is a convex loss. The function score for a test data point is defined as:

$$y = \frac{1}{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})}, \tag{3}$$

and the final classification decision is based on the sign of the function score. When the denominator is very close to 0, we can add a small value to it to make it numerically more stable.

Let $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_n) \in \mathbb{R}^{d\times n}$ denote the matrix for the unlabeled data currently available and the matrix $\mathbf{X} \in \mathbb{R}^{d\times t}$ denote the selected subset of unlabeled data for the oracle to label. So when no action has been taken, n is just equal to u which is the number of unlabeled data points to start with. On the other hand, while conventional active learning methods assume that $l > 0$, TED assumes that $l = 0$ and hence it is not designed to make use of discriminative information.

Problems (1) and (2) are equivalent as far as binary classification problems are concerned. From the derivation of TED, the covariance matrix of the estimation error of $\mathbf{w} - \mathbf{w}^\star$, where $\mathbf{w}^\star$ is the ground truth of $\mathbf{w}$, is proportional to the inverted Hessian matrix of $J(\mathbf{w})$:

$$\begin{aligned}\mathrm{cov}(\mathbf{w} - \mathbf{w}^\star) \propto \mathbf{C}_\mathbf{w} &= \Big(\frac{\partial^2 J(\mathbf{w})}{\partial\mathbf{w}\partial\mathbf{w}^T}\Big)^{-1}\\ &= \big(\boldsymbol{\phi}(\mathbf{X})\mathbf{Y}_\mathbf{X}^2\boldsymbol{\phi}(\mathbf{X})^T + \lambda\mathbf{I}_{d'}\big)^{-1},\end{aligned}$$

where $\mathbf{Y}_\mathbf{X}$ denotes a diagonal matrix whose diagonal elements are the function scores of the corresponding data points, $\mathbf{I}_d$ denotes the $d\times d$ identity matrix, $\boldsymbol{\phi}(\mathbf{X})$ denotes the data matrix of $\mathbf{X}$ after applying the feature map, and d' is the dimensionality of the data points after feature mapping. Then the predictive error on the whole unlabeled data set $\mathbf{V}$ has its covariance matrix proportional to $\mathbf{C}_\mathbf{f}$:

$$\begin{aligned}\mathbf{C}_\mathbf{f} =&\mathbf{Y}_\mathbf{V}\boldsymbol{\phi}(\mathbf{V})^T\mathbf{C}_\mathbf{w}\boldsymbol{\phi}(\mathbf{V})\mathbf{Y}_\mathbf{V}\\ =&\mathbf{Y}_\mathbf{V}\boldsymbol{\phi}(\mathbf{V})^T\Big(\boldsymbol{\phi}(\mathbf{X})\mathbf{Y}_\mathbf{X}^2\boldsymbol{\phi}(\mathbf{X})^T + \lambda\mathbf{I}_{d'}\Big)^{-1}\boldsymbol{\phi}(\mathbf{V})\mathbf{Y}_\mathbf{V}\\ =&-\frac{1}{\lambda}\mathbf{Y}_\mathbf{V}\boldsymbol{\phi}(\mathbf{V})^T\boldsymbol{\phi}(\mathbf{X})\mathbf{Y}_\mathbf{X}(\lambda\mathbf{I}_t + \mathbf{Y}_\mathbf{X}\boldsymbol{\phi}(\mathbf{X})^T\boldsymbol{\phi}(\mathbf{X})\mathbf{Y}_\mathbf{X})^{-1}\mathbf{Y}_\mathbf{X}\boldsymbol{\phi}(\mathbf{X})^T\boldsymbol{\phi}(\mathbf{V})\mathbf{Y}_\mathbf{V}\\ =&-\frac{1}{\lambda}\mathbf{Y}_\mathbf{V}\mathbf{K}_\mathbf{VX}\mathbf{Y}_\mathbf{X}(\lambda\mathbf{I}_t + \mathbf{Y}_\mathbf{X}\mathbf{K}_\mathbf{X}\mathbf{Y}_\mathbf{X})^{-1}\mathbf{Y}_\mathbf{X}\mathbf{K}_\mathbf{XV}\mathbf{Y}_\mathbf{V} + \frac{1}{\lambda}\mathbf{Y}_\mathbf{V}\mathbf{K}_\mathbf{V}\mathbf{Y}_\mathbf{V},\end{aligned}$$

where $\mathbf{Y}_\mathbf{V}$ is a diagonal matrix recording the function scores of the data points in $\mathbf{V}$, $\mathbf{K}_\mathbf{X}$ denotes the kernel matrix on $\mathbf{X}$, $\mathbf{K}_\mathbf{V}$ denotes the kernel matrix on $\mathbf{V}$, $\mathbf{K}_\mathbf{VX}$ denotes the kernel matrix between $\mathbf{V}$ and $\mathbf{X}$, and $\mathbf{K}_\mathbf{XV} = \mathbf{K}_\mathbf{VX}^T$. The last equality above holds as a result of the Woodbury identity.

We minimize the predictive variance by using the A-optimal design [15], which minimizes $\mathrm{tr}(\mathbf{C_f})$, the trace of $\mathbf{C_f}$, by treating $\mathrm{tr}(\mathbf{C_f})$ as a surrogate of the predictive variance. Since λ and $\mathbf{Y_V}\mathbf{K}_V\mathbf{Y_V}$ are constants, we define the optimization problem for DED as follows.

Definition 1. *Discriminative Experimental Design:*

$$\begin{aligned} \max_{\mathbf{X},\mathbf{Y_X}} \quad & \mathrm{tr}\Big[\mathbf{Y_V}\mathbf{K_{VX}}\mathbf{Y_X}(\lambda\mathbf{I}_t + \mathbf{Y_X}\mathbf{K_X}\mathbf{Y_X})^{-1}\mathbf{Y_X}\mathbf{K_{XV}}\mathbf{Y_V}\Big] \\ \textit{s.t.} \quad & \mathbf{X} \subset \mathbf{V}, |\mathbf{X}| = t, \mathbf{Y_X} \subset \mathbf{Y_V}. \end{aligned} \tag{4}$$

Here $\mathbf{X} \subset \mathbf{V}$ means the set of the columns in $\mathbf{X}$ is a subset of that in $\mathbf{V}$ and $|\mathbf{X}|$ denotes the number of data points in $\mathbf{X}$ which is just the number of columns in $\mathbf{X}$. Moreover, for diagonal matrices $\mathbf{Y_X}$ and $\mathbf{Y_V}$, $\mathbf{Y_X} \subset \mathbf{Y_V}$ means the set of the diagonal elements in $\mathbf{Y_X}$ is a subset of that in $\mathbf{Y_V}$.

Before we discuss how to solve problem (4) in the next subsection, let us first examine the relationship between DED and TED. We consider the linear case where

$$\mathbf{K_{VX}} = \mathbf{V}^T\mathbf{X},\ \mathbf{K_X} = \mathbf{X}^T\mathbf{X},\ \mathbf{K_{XV}} = \mathbf{X}^T\mathbf{V}.$$

Then the optimization problem for linear DED is

$$\begin{aligned} \max_{\mathbf{X},\mathbf{Y_X}} \quad & \mathrm{tr}\Big[\mathbf{Y_V}\mathbf{V}^T\mathbf{X}\mathbf{Y_X}(\lambda\mathbf{I}_t + \mathbf{Y_X}\mathbf{X}^T\mathbf{X}\mathbf{Y_X})^{-1}\mathbf{Y_X}\mathbf{X}^T\mathbf{V}\mathbf{Y_V}\Big] \\ \text{s.t.} \quad & \mathbf{X} \subset \mathbf{V}, |\mathbf{X}| = t, \mathbf{Y_X} \subset \mathbf{Y_V}. \end{aligned}$$

If we define $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{Y_X}$ and $\tilde{\mathbf{V}} = \mathbf{V}\mathbf{Y_V}$, then the optimization problem for linear DED becomes

$$\begin{aligned} \max_{\tilde{\mathbf{X}}} \quad & \mathrm{tr}\Big[\tilde{\mathbf{V}}^T\tilde{\mathbf{X}}(\lambda\mathbf{I}_t + \tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\tilde{\mathbf{V}}\Big] \\ \text{s.t.} \quad & \tilde{\mathbf{X}} \subset \tilde{\mathbf{V}}, |\tilde{\mathbf{X}}| = t, \end{aligned}$$

which is exactly the same as TED. So TED can be seen as a special case of DED with $\mathbf{Y_V} = \mathbf{I}_n$. Alternatively, we may regard DED as a weighted version of TED where the weights are related to the function scores of the data points. It is easy to see that both $\mathrm{tr}(\mathbf{C_f})$ and the objective function value of problem (4) do not depend on the signs of the function scores for all data points. So from the definition of function score given in (3), if a data point lies close to the decision boundary, its weight will be much larger than that of another point far from the boundary. A point which lies right at the decision boundary will have the largest weight. This is in line with the design criteria behind SVM active learning and batch mode active learning which use the decision function value as the uncertainty sampling criterion.

Similar to TED, linear DED also has a regularized least squares regression interpretation.

Theorem 1. *Linear Discriminative Experimental Design is equivalent to*

$$\begin{aligned} \min_{\mathbf{X},\mathbf{Y_X},\mathbf{A}} \quad & \sum_{i=1}^{n}\Big[\|y_i\mathbf{v}_i - \mathbf{X}\mathbf{Y_X}\mathbf{a}_i\|_2^2 + \lambda\|\mathbf{a}_i\|_2^2\Big] \\ \textit{s.t.} \quad & \mathbf{X} \subset \mathbf{V}, |\mathbf{X}| = t, \mathbf{Y_X} \subset \mathbf{Y_V} \\ & \mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \mathbb{R}^{t\times n}. \end{aligned}$$

The proof is similar to that of TED and hence we omit it here. From Theorem 1, we can find that, similar to TED, DED works by selecting representative data points after weighting them with the function scores. Thus DED can utilize both discriminative information and data distribution information to select the most informative data points to label.

3.2 Optimization Procedure

Even though the objective function of DED is similar to that of TED and so, in principle, we may use an optimization method similar to that in [9] to solve problem (4), here we choose to use a different optimization method which gives more insight into the nature of DED and TED.

Let $\phi(\mathbf{X})$ and $\phi(\mathbf{V})$ denote the data matrices after applying the feature map to each data point in $\mathbf{X}$ and $\mathbf{V}$, respectively. From these we get

$$\mathbf{K}_{XV} = \phi(\mathbf{X})^T\phi(\mathbf{V}),\ \mathbf{K}_X = \phi(\mathbf{X})^T\phi(\mathbf{X}),\ \mathbf{K}_V = \phi(\mathbf{V})^T\phi(\mathbf{V}).$$

Since $\phi(\mathbf{X})$ and $\mathbf{Y}_\mathbf{X}$ are submatrices of $\phi(\mathbf{V})$ and $\mathbf{Y}_\mathbf{V}$ respectively, we can define a selection indicator matrix $\mathbf{S} \in \{0,1\}^{n\times t}$ such that $\phi(\mathbf{X})\mathbf{Y}_\mathbf{X} = \phi(\mathbf{V})\mathbf{Y}_\mathbf{V}\mathbf{S}$. Because each column of $\phi(\mathbf{X})\mathbf{Y}_\mathbf{X}$ is from the column of $\phi(\mathbf{V})\mathbf{Y}_\mathbf{V}$, the (i,j)th element s_{ij} of $\mathbf{S}$ can be computed as

$$s_{ij} = \begin{cases} 1 & \text{if } (\phi(\mathbf{X})\mathbf{Y}_\mathbf{X})_{,j} \text{ is from } (\phi(\mathbf{V})\mathbf{Y}_\mathbf{V})_{,i} \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{M}_{,i}$ denotes the ith column of matrix $\mathbf{M}$. Since we need to select t data points from $\mathbf{V}$, one and only one element in each column of $\mathbf{S}$ is equal to 1. Moreover, since we want to select t distinct data points from $\mathbf{V}$, at most one element in each row of $\mathbf{S}$ is equal to 1. In other words, $\mathbf{S}$ consists of t distinct columns of the $n \times n$ identity matrix. So the columns of $\mathbf{S}$ consist of an orthogonal basis such that $\mathbf{S}^T\mathbf{S} = \mathbf{I}_t$. The constraint set for $\mathbf{S}$ can be defined as

$$C_S = \big\{\mathbf{S} \,|\, \mathbf{S} \in \{0,1\}^{n\times t}, \mathbf{S}^T\mathbf{1}_n = \mathbf{1}_t, \mathbf{S}\mathbf{1}_t \le \mathbf{1}_n\big\},$$

or equivalently

$$C_S = \big\{\mathbf{S} \,|\, \mathbf{S} \in \{0,1\}^{n\times t}, \mathbf{S}^T\mathbf{S} = \mathbf{I}_t\big\},$$

where $\mathbf{1}_m$ denotes an $m \times 1$ vector of all ones and $\le$ refers to the elementwise comparison between two vectors. Then we can get

$$\begin{aligned} \mathbf{Y}_\mathbf{V}\mathbf{K}_\mathbf{VX}\mathbf{Y}_\mathbf{X} &= \mathbf{Y}_\mathbf{V}\phi(\mathbf{V})^T\phi(\mathbf{X})\mathbf{Y}_\mathbf{X} \\ &= \mathbf{Y}_\mathbf{V}\phi(\mathbf{V})^T\phi(\mathbf{V})\mathbf{Y}_\mathbf{V}\mathbf{S} \\ &= \mathbf{Y}_\mathbf{V}\mathbf{K}_\mathbf{V}\mathbf{Y}_\mathbf{V}\mathbf{S} \\ \mathbf{Y}_\mathbf{X}\mathbf{K}_\mathbf{X}\mathbf{Y}_\mathbf{X} &= \mathbf{Y}_\mathbf{X}\phi(\mathbf{X})^T\phi(\mathbf{X})\mathbf{Y}_\mathbf{X} \\ &= \mathbf{S}^T\mathbf{Y}_\mathbf{V}\phi(\mathbf{V})^T\phi(\mathbf{V})\mathbf{Y}_\mathbf{V}\mathbf{S} \\ &= \mathbf{S}^T\mathbf{Y}_\mathbf{V}\mathbf{K}_\mathbf{V}\mathbf{Y}_\mathbf{V}\mathbf{S}. \end{aligned}$$

Thus the objective function in (4) becomes

$$\begin{aligned}\max_{\mathbf{S}} \quad & \mathrm{tr}\Big[(\lambda\mathbf{I}_t + \mathbf{S}^T\tilde{\mathbf{K}}_{\mathbf{V}}\mathbf{S})^{-1}\mathbf{S}^T\tilde{\mathbf{K}}_{\mathbf{V}}^2\mathbf{S}\Big] \\ \text{s.t.} \quad & \mathbf{S} \in C_S, \end{aligned} \tag{5}$$

where $\tilde{\mathbf{K}}_{\mathbf{V}} = \mathbf{Y}_{\mathbf{V}}\mathbf{K}_{\mathbf{V}}\mathbf{Y}_{\mathbf{V}}$. By imposing a constraint on $\mathbf{S}$ such that $\mathbf{S}^T\mathbf{S} = \mathbf{I}_t$, (5) can be rewritten as

$$\begin{aligned}\max_{\mathbf{S}} \ & \mathrm{tr}\Big[\Big(\mathbf{S}^T(\lambda\mathbf{I}_n + \tilde{\mathbf{K}}_{\mathbf{V}})\mathbf{S}\Big)^{-1}\mathbf{S}^T\tilde{\mathbf{K}}_{\mathbf{V}}^2\mathbf{S}\Big] \\ \text{s.t.} \ \ & \mathbf{S} \in C_S. \end{aligned} \tag{6}$$

The formulation of the objective function in (6) is identical to that of linear discriminant analysis (LDA) [18], so the optimal solution can be obtained by solving a generalized eigenvalue problem if there exist no constraints on $\mathbf{S}$.

Here we use a projection method to solve problem (6). That is, we first find the optimal solution of problem (6) while ignoring the constraints and then project the optimal solution found to the constraint set C_S.

We first solve the problem

$$\max_{\mathbf{S}} \ f(\mathbf{S}) = \Big[\Big(\mathbf{S}^T(\lambda\mathbf{I}_n + \tilde{\mathbf{K}}_{\mathbf{V}})\mathbf{S}\Big)^{-1}\mathbf{S}^T\tilde{\mathbf{K}}_{\mathbf{V}}^2\mathbf{S}\Big]. \tag{7}$$

According to the analysis in [18], the optimal solution $\mathbf{S}^\star$ consists of the top t eigenvectors of $(\lambda\mathbf{I}_n + \tilde{\mathbf{K}}_{\mathbf{V}})^{-1}\tilde{\mathbf{K}}_{\mathbf{V}}^2$. Let $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ and $\boldsymbol{\Pi} = \mathrm{diag}(\pi_1, \ldots, \pi_n)$ denote the eigenvectors and eigenvalues, respectively, of the matrix $\tilde{\mathbf{K}}_{\mathbf{V}}$ where $\pi_1 \geq \ldots \geq \pi_n$. Then, by using the fact that $\mathbf{Q}$ is an $n \times n$ orthogonal matrix because $\mathbf{Q}$ is the eigenvector matrix of a symmetric matrix $\tilde{\mathbf{K}}_{\mathbf{V}}$, we can get

$$\begin{aligned}(\lambda\mathbf{I}_n + \tilde{\mathbf{K}}_{\mathbf{V}})^{-1}\tilde{\mathbf{K}}_{\mathbf{V}}^2 &= (\lambda\mathbf{I}_n + \mathbf{Q}^T\boldsymbol{\Pi}\mathbf{Q})^{-1}\mathbf{Q}^T\boldsymbol{\Pi}^2\mathbf{Q} \\ &= (\lambda\mathbf{Q}^T\mathbf{Q} + \mathbf{Q}^T\boldsymbol{\Pi}\mathbf{Q})^{-1}\mathbf{Q}^T\boldsymbol{\Pi}^2\mathbf{Q} \\ &= \mathbf{Q}^T(\lambda\mathbf{I}_n + \boldsymbol{\Pi})^{-1}\mathbf{Q}\mathbf{Q}^T\boldsymbol{\Pi}^2\mathbf{Q} \\ &= \mathbf{Q}^T(\lambda\mathbf{I}_n + \boldsymbol{\Pi})^{-1}\boldsymbol{\Pi}^2\mathbf{Q}.\end{aligned}$$

So $\mathbf{q}_i$ is an eigenvector of $(\lambda\mathbf{I}_n + \tilde{\mathbf{K}}_{\mathbf{V}})^{-1}\tilde{\mathbf{K}}_{\mathbf{V}}^2$ with the corresponding eigenvalue as

$$\pi_i' = h(\pi_i) = \pi_i^2/(\lambda + \pi_i).$$

Then $\mathbf{S}^\star$ consists of t eigenvectors with the t largest eigenvalues in $\{\pi_i'\}$. We find that $h(x)$ is monotonically increasing for $x \geq 0$ since $h'(x) = \frac{x^2+2\lambda x}{(x+\lambda)^2} \geq 0$ given $\lambda > 0$, and $h(x)$ is strictly increasing when $x > 0$. So we have $\pi_i' > \pi_j'$ when $\pi_i > \pi_j \geq 0$ and $\mathbf{S}^\star$ consists of the top t eigenvectors of $\tilde{\mathbf{K}}_{\mathbf{V}}$.

We next project $\mathbf{S}^\star$ to the set C_S. Note that $\mathbf{S}^\star$ is not a unique optimal solution of problem (7) since for any orthogonal matrix $\mathbf{P} \in \mathbb{R}^{t\times t}$ we have $f(\mathbf{S}^\star\mathbf{P}) = f(\mathbf{S}^\star)$. So we define the objective function for the projection as

$$\begin{aligned}\min_{\mathbf{P},\mathbf{Q}} \ & \|\mathbf{S}^\star\mathbf{P} - \mathbf{Q}\|_F^2 \\ \text{s.t.} \ \ & \mathbf{Q} \in C_S, \ \mathbf{P}\mathbf{P}^T = \mathbf{I}_t, \end{aligned} \tag{8}$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm. We simplify the objective function as

$$\begin{aligned}\|\mathbf{S}^\star\mathbf{P}-\mathbf{Q}\|_F^2 &= \mathrm{tr}\Big((\mathbf{S}^\star\mathbf{P}-\mathbf{Q})^T(\mathbf{S}^\star\mathbf{P}-\mathbf{Q})\Big)\\ &= \mathrm{tr}(\mathbf{Q}^T\mathbf{Q})+\mathrm{tr}(\mathbf{P}^T(\mathbf{S}^\star)^T\mathbf{S}^\star\mathbf{P})-2\mathrm{tr}(\mathbf{Q}^T\mathbf{S}^\star\mathbf{P})\\ &= 2t-2\,\mathrm{tr}(\mathbf{Q}^T\mathbf{S}^\star\mathbf{P}).\end{aligned}$$

Note that the last equality holds because $\mathbf{Q}^T\mathbf{Q}=\mathbf{I}_t$, $(\mathbf{S}^\star)^T\mathbf{S}^\star=\mathbf{I}_t$ and $\mathbf{P}^T\mathbf{P}=\mathbf{I}_t$. So minimizing $\|\mathbf{S}^\star\mathbf{P}-\mathbf{Q}\|_F^2$ is equivalent to maximizing $\mathrm{tr}(\mathbf{Q}^T\mathbf{S}^\star\mathbf{P})$ and so problem (8) is equivalent to the following problem

$$\begin{aligned}&\max_{\mathbf{P},\mathbf{Q}}\ \mathrm{tr}(\mathbf{Q}^T\mathbf{S}^\star\mathbf{P})\\ &\ \text{s.t.}\ \ \mathbf{Q}\in C_S,\ \mathbf{P}\mathbf{P}^T=\mathbf{I}_t.\end{aligned}\tag{9}$$

However, problem (9) is not convex. Here we use an alternating method to solve it. Specifically, we first find the optimal solution with respect to $\mathbf{Q}$ when $\mathbf{P}$ is fixed and then find the optimal solution with respect to $\mathbf{P}$ when $\mathbf{Q}$ is fixed.

When $\mathbf{P}$ is fixed, the optimization problem with respect to $\mathbf{Q}$ is

$$\begin{aligned}&\max_{\mathbf{Q}}\,\mathrm{tr}(\mathbf{Q}^T\mathbf{S}^\star\mathbf{P})\\ &\ \text{s.t.}\ \mathbf{Q}\in\{0,1\}^{n\times t},\mathbf{Q}^T\mathbf{1}_n=\mathbf{1}_t,\mathbf{Q}\mathbf{1}_t\le\mathbf{1}_n.\end{aligned}\tag{10}$$

This problem is to find the t largest elements in $\mathbf{S}^\star\mathbf{P}$ where no two elements can be in the same column or the same row. This is an integer programming problem with no efficient solution. Based on our observation that the largest elements of different columns in $\mathbf{S}^\star$ usually lie in different rows, we propose a greedy algorithm for the problem: we first find the largest element in $\mathbf{S}^\star\mathbf{P}$ (if there exist multiple elements that are the largest, we can choose any one of them) and mark its row and column; then from the unmarked columns and rows we find the largest one and also mark it; this procedure is repeated until we find t elements.

When $\mathbf{Q}$ is fixed, the optimization problem with respect to $\mathbf{P}$ is

$$\begin{aligned}&\max_{\mathbf{P}}\ \mathrm{tr}(\mathbf{Q}^T\mathbf{S}^\star\mathbf{P})\\ &\ \text{s.t.}\ \ \mathbf{P}\mathbf{P}^T=\mathbf{I}_t.\end{aligned}\tag{11}$$

We define a Lagrangian using a symmetric matrix multiplier $\boldsymbol{\Lambda}$ as

$$L(\mathbf{P},\boldsymbol{\Lambda})=\mathrm{tr}(\mathbf{Q}^T\mathbf{S}^\star\mathbf{P})-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Lambda}(\mathbf{P}\mathbf{P}^T-\mathbf{I}_t)).$$

Then the optimal solution $(\mathbf{P}^\star,\boldsymbol{\Lambda}^\star)$ satisfies

$$\frac{\partial L}{\partial\mathbf{P}}=(\mathbf{S}^\star)^T\mathbf{Q}-\boldsymbol{\Lambda}^\star\mathbf{P}^\star=0$$

which leads to $\boldsymbol{\Lambda}^\star=(\mathbf{S}^\star)^T\mathbf{Q}(\mathbf{P}^\star)^T$ by right-multiplying $(\mathbf{P}^\star)^T$. Utilizing the fact that $\mathbf{P}^\star$ is a $t\times t$ orthogonal matrix, we get

$$\boldsymbol{\Lambda}^\star(\boldsymbol{\Lambda}^\star)^T=(\mathbf{S}^\star)^T\mathbf{Q}\mathbf{Q}^T\mathbf{S}^\star.$$

Let $(\mathbf{S}^\star)^T\mathbf{Q} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{R}^T$ be the singular value decomposition (SVD) where $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{R} \in \mathbb{R}^{t\times t}$. So

$$\boldsymbol{\Lambda}^\star(\boldsymbol{\Lambda}^\star)^T = \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^T$$

and

$$\boldsymbol{\Lambda}^\star = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T$$

since $\boldsymbol{\Lambda}^\star$ is symmetric. Then we can get the optimal $\mathbf{P}^\star$ as

$$\mathbf{P}^\star = (\boldsymbol{\Lambda}^\star)^{-1}(\mathbf{S}^\star)^T\mathbf{Q} = \mathbf{U}\mathbf{R}^T.$$

The main computational cost includes computing the eigenvectors of $\tilde{\mathbf{K}}$ corresponding to the largest t eigenvalues only one time which costs $O(n^2t)$ and SVD for $(\mathbf{S}^\star)^T\mathbf{Q}$ which costs $O(t^3)$. So the computational cost of our method is $O(n^2t)$, which is more efficient than that of [9] with $O(n^3)$ complexity. Moreover, our method provides a better characterization of the nature of DED. The regularization parameter λ has significant effect on the optimization procedure of TED in [9] but DED seems to be insensitive to it. This property is desirable because DED is robust against λ.

4 Experiments

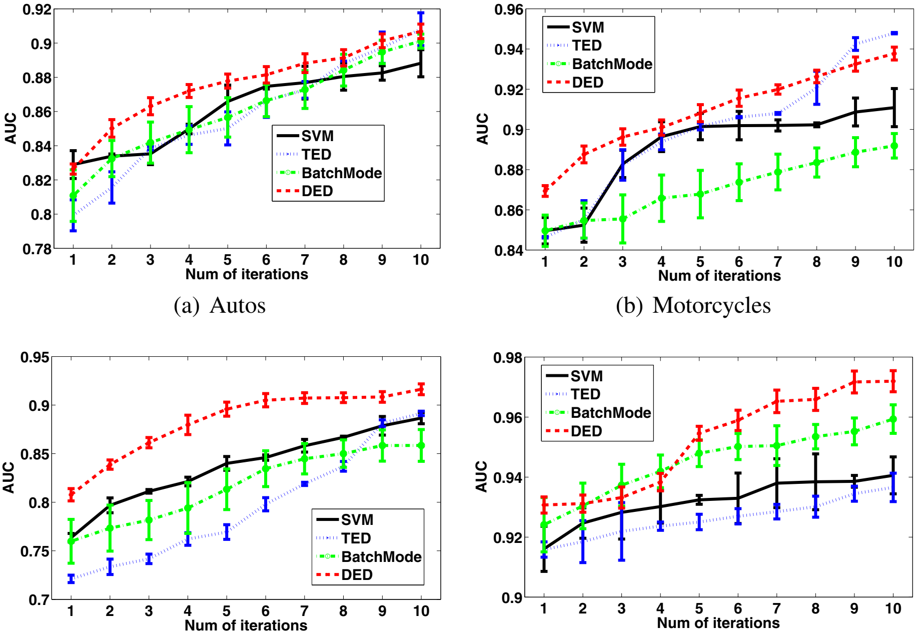
In this section, we study DED empirically and compare its performance with several active learning methods, which include TED, SVM active learning and batch mode active learning [19].

We conduct experiments on two public benchmark data sets. The first one is a subset of the Newsgroups corpus [20], which consists of 3970 documents with TFIDF features of 8014 dimensions. Each document belongs to exactly one of four categories: autos, motorcycles, baseball and hockey. The other one is the Reuters data set, which is a subset of the RCV1-v2 data set [21]. Each document in the Reuters data set belongs to at least one of four categories: CCAT, ECAT, GCAT and MCAT.

In the experiments, we simply treat the multi-class/label classification problem as a set of binary classification problems by using the one-versus-all scheme, i.e., documents from the target category are labeled as positive examples and those from the other categories are labeled as negative examples. We use area under the ROC curve (AUC) as the performance measure to measure the overall classification performance, because in our setting, each binary classification task is unbalanced (only about 25% of the documents in the Newsgroups data set and about 30% of the documents in the Reuters data set are positive).

In our experiments, t is set to 5 and all the regularization parameters in DED, TED and SVM active learning are set to 0.01. We initially have five labeled data points for each class before active learning starts.

We first test our method on the Newsgroups data set. The AUC values over the four binary classification tasks are reported in Figure 1(a) to 1(d). From the results, we can see that on Autos, Motorcycles and Baseball, DED outperforms the other methods in the early stage. This observation validates the contribution of data distribution information. When the labeled data is scarce, data distribution information may be more important than discriminative information since the estimated decision boundary in this stage is not very accurate.



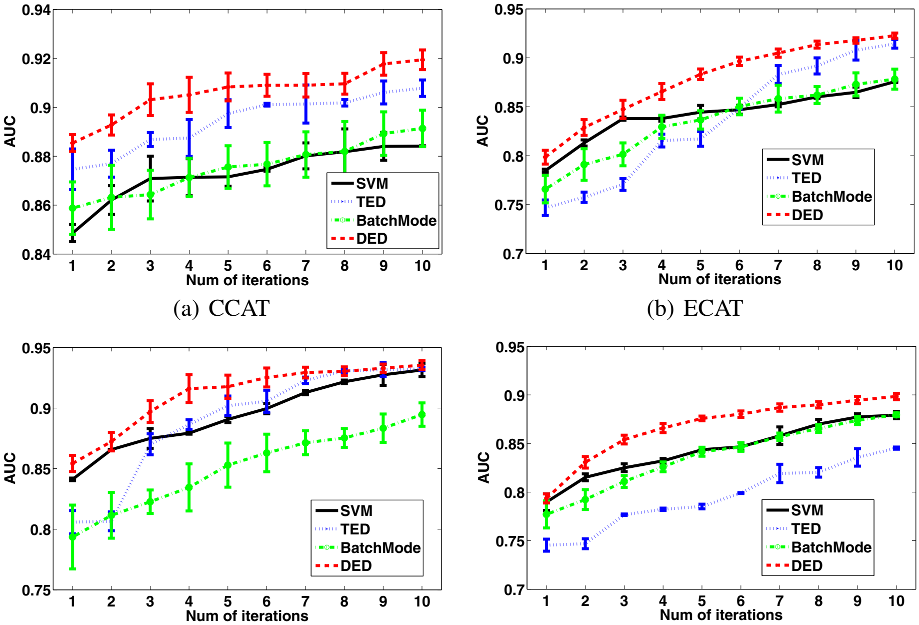
(a) Autos

(b) Motorcycles

(c) Baseball

(d) Hockey

Fig. 1. Learning curves for four binary classification tasks on Newsgroups data



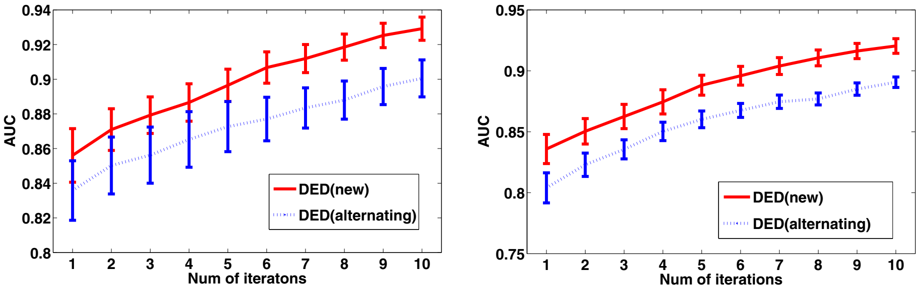
(a) CCAT

(b) ECAT

(c) GCAT

(d) MCAT

Fig. 2. Learning curves for four binary classification tasks on Reuters data



(a) Newsgroups data (b) Reuters data

Fig. 3. Comparison of two optimization methods for DED on the two data sets. DED(new) uses the optimization procedure proposed in our paper and DED(alternating) utilizes the alternating optimization method proposed in [9].

We now compare the four methods on the Reuters data set. The AUC values over the four tasks are reported in Figure 2(a) to 2(d). For the four categories, DED consistently outperforms the second best by a large margin. This observation validates the contribution of discriminative information to experimental design.

Moreover, to see the effect of the optimization method proposed in section 3.2, we compare the performance of DED when using our proposed optimization method and the one proposed in [9]. The AUC values averaged over four binary classification tasks of the two data sets are reported in Fig. 3(a) and Fig. 3(b). We can see that the performance of our proposed method is better than the one proposed in [9].

5 Conclusion

We have proposed in this paper a novel active learning method which integrates margin-based discriminative information and data distribution information to define the unlabeled data selection criterion. As the next step to extend this work further, we will investigate the integration of active learning and semi-supervised learning to further improve the performance by exploiting unlabeled data.

Acknowledgment. This research has been supported by General Research Fund 622209 from the Research Grants Council of Hong Kong.

References

1. Chapelle, O., Zien, A., Schölkopf, B. (eds.): Semi-Supervised Learning. MIT Press, Boston (2006)
2. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine Learning 15, 201–221 (1994)
3. Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. Journal of Artificial Intelligence Research 4, 129–145 (1996)

4. Campbell, C., Cristianini, N., Smola, A.J.: Query learning with large margin classifiers. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 111–118. Stanford University, Standord (2000)
5. Schohn, G., Cohn, D.: Less is more: Active learning with support vector machines. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 839–846. Stanford University, Standord (2000)
6. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 999–1006. Stanford University, Standord (2000)
7. Hoi, S.C.H., Jin, R., Zhu, J., Lyu, M.R.: Batch mode active learning and its application to medical image classification. In: Proceedings of the Twenty-Third International Conference on Machine Learning, Pittsburgh, Pennsylvania, USA, pp. 417–424 (2006)
8. Guo, Y., Schuurmans, D.: Discriminative batch mode active learning. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems, vol. 20, pp. 593–600 (2007)
9. Yu, K., Bi, J., Tresp, V.: Active learning via transductive experimental design. In: Proceedings of the Twenty-Third International Conference on Machine Learning, Pittsburgh, Pennsylvania, USA, pp. 1081–1088 (2006)
10. Sindhwani, V., Melville, P., Lawrence, R.D.: Uncertainty sampling and transductive experimental design for active dual supervision. In: Proceedings of the 26th International Conference on Machine Learning, Montreal, Quebec, Canada, pp. 953–960 (2009)
11. Lewis, D.D., Catlett, J.: Heterogeneous uncertainty sampling for supervised learning. In: Proceedings of the 11th International Conference on Machine Learning, pp. 148–156. Morgan Kaufmann, San Francisco (1994)
12. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the 5th Annual Workshop on Computational Learning Theory, New York, NY, USA, 287–294 (1992)
13. Nguyen, H.T., Smeulders, A.: Active learning using preclustering. In: Proceedings of the 21st International Conference on Machine learning, Banff, Alberta, Canada (2004)
14. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: Proceedings of the 18th International Conference on Machine Learning, San Francisco, CA, USA, pp. 441–448 (2001)
15. Atkinson, A.C., Donev, A.N. (eds.): Optimum Experiment Designs. Oxford University Press, Boston (1992)
16. Gestel, T.V., Suykens, J.A.K., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., Moor, B.D., Vandewalle, J.: Benchmarking least squares support vector machine classifiers. Machine Learning 54(1), 5–32 (2004)
17. 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 Twenty-Fifth International Conference on Machine Learning, Helsinki, Finland, pp. 408–415 (2008)
18. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press, New York (1991)
19. Hoi, S.C.H., Jin, R., Zhu, J., Lyu, M.R.: Semi-supervised svm batch mode active learning for image retrieval. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska, USA (2008)
20. Yu, K., Zhu, S., Xu, W., Gong, Y.: Non-greedy active learning for text categorization using convex ansductive experimental design. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Singapore, pp. 635–642 (2008)
21. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: RCV1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361–397 (2004)

Active Learning with Evolving Streaming Data

Indrė Žliobaitė[1,2], Albert Bifet[2], Bernhard Pfahringer[2], and Geoff Holmes[2]

[1] Bournemouth University, Poole, UK
izliobaite@bournemouth.ac.uk
[2] University of Waikato, Hamilton, New Zealand
{abifet,geoff,bernhard}@cs.waikato.ac.nz

Abstract. In learning to classify streaming data, obtaining the true labels may require major effort and may incur excessive cost. Active learning focuses on learning an accurate model with as few labels as possible. Streaming data poses additional challenges for active learning, since the data distribution may change over time (concept drift) and classifiers need to adapt. Conventional active learning strategies concentrate on querying the most uncertain instances, which are typically concentrated around the decision boundary. If changes do not occur close to the boundary, they will be missed and classifiers will fail to adapt. In this paper we develop two active learning strategies for streaming data that explicitly handle concept drift. They are based on uncertainty, dynamic allocation of labeling efforts over time and randomization of the search space. We empirically demonstrate that these strategies react well to changes that can occur anywhere in the instance space and unexpectedly.

1 Introduction

Supervised learning models the relationship between the observed variables of an instance and the target variable (label). To build a predictor we need to know the true labels of the training data. Often unlabeled data is abundant but labeling is expensive. Labels can be costly to obtain due to required human input (labor cost). Consider, for example, textual news arriving as a stream. The goal is to predict if a news item will be interesting to a given user at a given time. The interests of the user may change. To obtain training data the historical news needs to be read and labeled as interesting or not interesting. This requires human labor. For instance, Amazon Mechanical Turk[1] provides a marketplace for intelligent human labeling. Labeling can also be costly due to a required expensive, intrusive or destructive laboratory test. Consider a production process in a chemical plant where the goal is to predict the quality of production output. The relationship between input and output quality might change over time due to constant manual tuning, complementary ingredients or replacement of physical sensors. In order to know the quality of the output (the true label) a laboratory test needs to be performed which is costly. Under such conditions it may be unreasonable to require true labels for all incoming instances.

[1] https://www.mturk.com

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 597–612, 2011.

Active learning studies how to label selectively instead of asking for all true labels. It has been extensively studied in pool-based [14] and online settings [6]. In pool-based settings the decision concerning which instances to label is made from all historical data. In this paper we explore active learning in data stream settings, where this decision needs to be made immediately for every incoming instance, as there is no re-access to it. The main difference between online active learning and active learning in data streams is in expectations around changes. Online active learning typically fixes a threshold (e.g. an uncertainty threshold) and asks for the true label if the threshold is exceeded. In data streams the relationship between the input data and the label may change (concept drift) and these changes can happen anywhere in the instance space. Thus, existing active learning strategies may never query instances from some regions and thus may never know that changes are happening and therefore never adapt. Moreover, in data streams we cannot keep the decision threshold or a region of uncertainty fixed, as eventually the system would stop learning and fail to react to changes. Finally, active learning with data streams must preserve the incoming data distribution to the extent that changes could be detected as they happen.

We study active learning strategies specifically for data streams. In brief, the setting is as follows. Data arrives in a stream, and predictions need to be made in real time. Concept drift is expected, thus learning needs to be adaptive. The true label can be requested immediately or never, as the instances are regularly discarded from memory. Our goal is to maximize prediction accuracy over time, while keeping the labeling costs fixed within an allocated budget. After scanning an instance and outputting the prediction for it, we need a strategy to decide, whether or not to query for the true label so that our model could train itself with this new instance. Regular retraining is needed due to changes in data distribution. Active learning strategies in data streams in addition to being able to learn an accurate classifier in stationary situations, need to be able to

- balance the labeling budget over time;
- notice changes happening anywhere in the instance space;
- preserve the distribution of the incoming data for detecting changes;

In this paper we develop two such strategies, assuming that the adaptive learning technique is externally given. Experimental evaluation on real data streams demonstrates that the proposed approaches effectively handle concept drift while saving labeling costs as we do not need to label every instance. To the best of our knowledge this study is the first to address active learning for instance-incremental streaming data (we preclude methods that learn from a stream in batches) where historical data cannot be stored in memory.

The paper is organized as follows. Section 2 discusses related work. Section 3 presents our active learning strategies for data streams. In Section 4 we analytically investigate the properties of the proposed strategies. Section 5 presents experimental evaluation with real streaming data. Section 6 concludes the study.

2 Related Work

Online active learning has been a subject of a number of studies, where the data distribution is assumed to be static [2, 6, 10, 19]. As discussed in the introduction, static online active learning is not designed to handle changes. As we will demonstrate in our analysis and experiments, existing strategies are able to handle drifts if changes happen to be gradual and close to the current decision boundary; however, the reaction to change might be slow. When changes happen far from the decision boundary, such methods fail completely. Those situations require advanced strategies, we develop several such strategies in this study.

The problem of label availability in data streams with concept drift has been acknowledged in several recent works [11, 13, 21, 22]. Most convert a stationary active learning or a semi-supervised learning method to an online setting by partitioning a stream into batches and applying stationary strategies within each batch. These works differ from our study in two major aspects. First, their strategies require to inspect a batch of instances at a time, thus they need to assume that limited re-access to data is possible. In contrast, our stream setting requires to make labeling decisions online at the time of scanning each instance.

Moreover, the existing active learning or semi-supervised learning strategies only complement concept drift handling strategies, they are not tailored to handle concept drift directly. It is assumed, that the data within a batch is stationary and the goal is to train an accurate classifier, while minimizing labeling costs. Adaptation and drift detection is separate. Thus, these techniques help to learn an accurate current model with fewer labels, but they are not designed to adapt to changes faster or with fewer labels. If changes happen far from the decision boundary, they are likely to be missed. In contrast, we tailor active learning to handle concept drift directly online and search the instance space explicitly.

Parts of some of the literature are conceptually related to our approach. We highlight these in the remaining part of this section.

The first group uses active learning strategies [16, 17, 24] with batches. Zhu et al. [24] build a classifier on a small portion of data within a batch at random and use uncertainty sampling to label more instances within this batch. A new classifier in each batch is needed to take into account concept drift. Similarly, Masud et al. [17] use uncertainty sampling within a batch to request labels. In addition, they use the unlabeled instances with their predicted labels for training. Lindstrom et al. [16] use uncertainty sampling to label the most representative instances within each new batch. They do not explicitly detect changes, instead they use a sliding window approach, which discards the oldest instances. In summary, these approaches apply static active learning to batches, which is not possible in data streams where historical instances cannot be stored in memory.

Note that typically (a real) concept drift refers to changes in the posterior distributions of data $p(y|X)$, where X contains the observed variables and y is the corresponding target variable. In other words, real concept drift means that the unlabeled data distribution does not change, only the class conditional distributions change. In contrast, data evolution refers to changes in the unconditional distribution of data $p(X)$.

A few works integrate active learning and change detection [7, 11] in the sense that they first detect change and only if change is detected do they ask for representative true labels. However, only a change in $p(X)$ can be handled this way. We address real concept drift, which means that $p(y|X)$ changes and these changes cannot be detected without labels. Thus, these works are not solving the same problem as our approaches. Additionally, they use pool-based strategies.

Another group of works uses semi-supervised learning approaches to label some of the unlabeled data automatically [13, 21, 22]. Klinkenberg [13] separates changes in $p(X)$ from changes in $p(y|X)$ and uses existing semi-supervised techniques to label when $p(X)$ drifts, while a separate non active learning strategy handles changes in $p(y|X)$. Widyantoro and Yen [21] first verify that a concept is stable and only then apply semi-supervised techniques for labeling. Woolam et al. [22] first label some instances within a batch at random and then propagate those labels to other instances using micro clustering. In both works automated labeling concerns only the subsets of the learning problem, which are assumed to be stationary (no real concept drift), which does not correspond to data stream settings we are addressing, thus they are not directly comparable.

Two related studies [4, 23] address a slightly different problem, they assume that only a part of the data in a stream is labeled and propose a method to learn from both labeled and unlabeled data. The problem setting is also different from that of this paper, as they do not perform active learning (active labeling).

A few works are related to the aspect of variable active learning criterion, which we introduce as a part of our strategies. Attenberg and Provost [2] introduce active inference as an additional aspect of online active learning. They maintain a budget per time period in a stream setting, while instead of uncertainty they estimate the utility of labeling an instance, which also takes into account the expected frequency of seeing a particular instance again. It assumes a possibility of repeated examples. The labeling threshold is fixed, but it depends on more than just uncertainty. This work is limited to the stationary setting.

Cesa-Bianchi et al [5] develop an online active learning method for a perceptron based on selective sampling using a variable labeling threshold $b/(b+|p|)$, where b is a parameter and p is the prediction of the perceptron. The threshold itself is based on certainty expectations, while the labels are queried at random. This mechanism could allow adaptation to changes, although they did not explicitly consider concept drift.

3 Strategies

In this Section we present active learning strategies for data streams. We start with two basic techniques and discuss their drawbacks. Then we introduce our strategies in two steps, where each step aims to overcome a challenge posed by the data stream setting. We start with a formal definition of our setting.

3.1 Setting

Let X_t be an instance, y_t its true label, where t indicates the time when an instance arrives. $X_1, X_2, \ldots, X_t, \ldots$ is then a data stream. The labeling cost is the same for any instance. We impose a budget B to obtain the true labels, which is expressed as a fraction of the number of incoming instances. $B = 1$ means that all arriving instances are labeled, whereas $B = 0.2$ means that 20% of the arriving instances are labeled.

Figure 1 shows our framework, that combines active learning strategies with adaptive learning. In this work we use the change detection technique of [8]: when the accuracy of the classifier begins to decrease a new classifier is built and trained with new incoming instances. When a change is detected, the old classifier is replaced by the new one.

```
ACTIVE LEARNING FRAMEWORK
  Input: labeling budget $B$ and strategy parameters

  for each $X_t$ - incoming instance,
      do if ACTIVE LEARNING STRATEGY$(X_t, B, \ldots)$ = true
          then request the true label $y_t$ of instance $X_t$
              train classifier $L$ with $(X_t, y_t)$
              if $L_n$ exists then train classifier $L_n$ with $(X_t, y_t)$
              if change warning is signaled
                then start a new classifier $L_n$
              if change is detected
                then replace classifier $L$ with $L_n$
```

Fig. 1. Strategy framework

3.2 Random Strategy

The first (baseline) strategy is naive in the sense that it labels the incoming instances at random instead of actively deciding which label would be more relevant. For every incoming instance the true label is requested with a probability B, where B is the budget. See Figure 2 for a formal description.

3.3 Fixed Uncertainty Strategy

Uncertainty sampling is perhaps the simplest and the most common active learning strategy [20]. The idea is to label the instances for which the current classifier is the least confident. In an online setting it corresponds to labeling the instances for which the certainty is below some fixed threshold. A simple way to measure uncertainty is to use the posterior probability estimates, output by a classifier. The uncertainty strategy with a fixed threshold is presented in Figure 3.

```
Random($X_t, B$)
    Input: $X_t$ - incoming instance, $B$ -labeling budget.
    Output: *label* $\in \{\mathbf{true}, \mathbf{false}\}$ indicates whether to request the true label $y_t$.

generate a uniform random variable $\xi_t \in [0, 1]$
**return** $\xi_t < B$
```

Fig. 2. Random strategy

```
FixedUncertainty($X_t, \theta, L$)
    Input: $X_t$ - incoming instance , $\theta$ - labeling threshold, $L$ - trained classifier.
    Output: *label* $\in \{\mathbf{true}, \mathbf{false}\}$ indicates whether to request the true label $y_t$.

$\hat{y_t} = \arg\max_y P_L(y|X_t)$, where $y \in \{1, \ldots, c\}$ is one of the class labels.
**return** $P_L(\hat{y_t}|X_t) < \theta$
```

Fig. 3. Fixed uncertainty strategy

3.4 Variable Uncertainty Strategy

One of the challenges with the uncertainty strategy in a streaming data setting is how to distribute the labeling effort over time. If we use a fixed threshold after some time a classifier would either exhaust its budget or reach the threshold certainty. In both cases it will stop learning and thus fail to adapt to changes.

Instead of labeling the instances that are less certain than the threshold we would like to label the least certain instances within a time interval. Thus we introduce *a variable threshold*, which adjusts itself depending on the incoming data to align with the budget. If a classifier becomes more certain (stable situations), the threshold expands to be able to capture the most uncertain instances. If a change happens and suddenly a lot of labeling requests appear, then the threshold is contracted to query the most uncertain instances first.

It may seem counter intuitive that we are asking for more labels at certain situations and fewer labels at changes. In fact, our dynamic threshold assures that we are asking for the same number of labels in all situations. This is how we balance the budget as we do not know when or how often changes will be happening, so we aim to spend the budget uniformly over time.

The uncertainty strategy with a variable threshold is described in Figure 4.

3.5 Uncertainty Strategy with Randomization

The uncertainty strategy always labels the instances that are close to the decision boundary of the classifier. In data streams changes may happen anywhere in the instance space. When concept drift happens in labels the classifier will not notice it without the true labels. In order not to miss concept drift we would

```
VariableUncertainty($X_t, L, B, s$)
    Input: $X_t$ - incoming instance , $L$ trained classifier, $B$ - budget, $s$ - adjusting step.
    Output: $label \in \{\textbf{true}, \textbf{false}\}$ indicates whether to request the true label $y_t$.
    Starting defaults: total labeling cost $u = 0$, initial labeling threshold $\theta = 1$.

    if ($u/t < B$)
       then budget is not exceeded,
            $\hat{y_t} = \arg\max_y P_L(y|X_t)$, where $y \in \{1, \ldots, c\}$ is one of the class labels.
            if ($P_L(\hat{y_t}|X_t) < \theta$)
               then uncertainty below the threshold
                    $u = u + 1$ labeling costs increase,
                    $\theta = \theta(1 - s)$ the threshold decreases,
                    return true
               else certainty is good
                    $\theta = \theta(1 + s)$ make the uncertainty region wider.
                    return false
       else budget is exceeded
            return false
```

Fig. 4. Uncertainty strategy with a dynamic threshold

```
VariableRandomizedUncertainty($X_t, L, B, s, \delta$)
    Input: $X_t$ - incoming instance , $L$ trained classifier, $B$ - budget, $s$ - adjusting step,
           $\delta$ - variance of the threshold randomization.
    Output: $label \in \{\textbf{true}, \textbf{false}\}$ indicates whether to request the true label $y_t$.
    Starting defaults: total labeling cost $u = 0$, initial labeling threshold $\theta = 1$.

    if ($u/t < B$)
       then budget is not exceeded,
            $\hat{y_t} = \arg\max_y P_L(y|X_t)$, where $y \in \{1, \ldots, c\}$ is one of the class labels.
            $\theta_{randomized} = \theta \times \eta$, where $\eta \in \mathcal{N}(1, \delta)$ is a random multiplier,
            if ($P_L(\hat{y_t}|X_t) < \theta_{randomized}$)
               then uncertainty below the threshold
                    $u = u + 1$ labeling costs increase,
                    $\theta = \theta(1 - s)$ the threshold decreases,
                    return true
               else certainty is good
                    $\theta = \theta(1 + s)$ make the uncertainty region wider
                    return false
       else budget is exceeded
            return false
```

Fig. 5. Uncertainty strategy with randomization

like, from time to time, to label the instances about which the classifier is very certain. For that purpose for every instance we randomize the labeling threshold by multiplying by a normally distributed random variable that follows $\mathcal{N}(1, \delta)$. This way we will label the instances that are close to the decision boundary more often, but occasionally we will also label some distant instances.

Table 1. Summary of strategies

	Controlling Budget	Instance space Coverage	Labeled Data Distribution
Random	present	full	iid
Fixed uncertainty	no	fragment	biased
Variable uncertainty	handled	fragment	biased
Randomized uncertainty	handled	full	biased

This strategy trades off labeling some very uncertain instances for labeling very certain instances, in order not to miss changes. Thus, in stationary situations this strategy is expected to perform worse than the uncertainty strategy, but in changing situations it is expected to adapt faster. The uncertainty strategy with randomization is presented in Figure 5.

Table 1 summarizes the four strategies with respect to the requirements indicated in the introduction. The random strategy satisfies all three requirements. Randomized uncertainty satisfies budget and coverage, but it produces biased labeled data. The variable uncertainty satisfies only budget and the fixed uncertainty satisfies none.

4 Analysis of How the Labeling Strategies Learn

In this section we explore the main learning aspects of the strategies: the ability to notice changes in dynamic situations and to learn accurate classifiers in stationary situations. In order to demonstrate the behavior of the strategies in controlled settings we employ synthetic data in $2D$. The data is distributed uniformly at random in a square, the distribution $p(X)$ does not change over time, $p(y|X)$ changes. This data represents a binary classification problem. The initial decision boundary is set at $x_1 = x_2$, as illustrated in Figure 6 (left).

Figure 7 shows how the strategies work on the hyperplane problem. The instances that would be labeled by different strategies are visualized. Each strategy labels the same number of instances. The random strategy labels uniformly from the instance space, while the uncertainty strategy concentrates around the decision boundary. The randomized uncertainty infuses randomization into the uncertainty sampling to cover the full instance space.

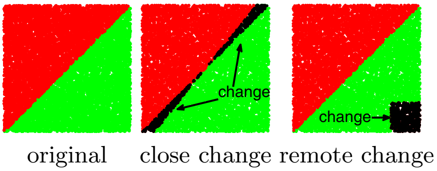


Fig. 6. Data with changes close and far from the decision boundary

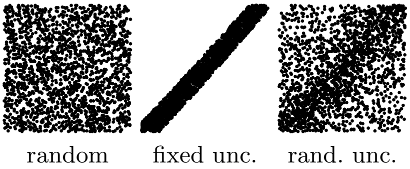


Fig. 7. 20% of the true labels queried with different labeling strategies

4.1 Ability to Learn Changes

Let us look at how concept drift is handled by our strategies. We investigate two situations: a change happening close to the decision boundary, and a remote change. Figure 6 (center and right) presents in black the regions in the instance space that are affected by a change. The center plot illustrates a change that happens close to the decision boundary. The right plot illustrates a remote change. In both examples the number of instances that change is the same.

We analyze how well our strategies would notice those changes. Figure 8 (left and center) plots the proportion of the changed (black) instances queried by each strategy. The plots can be interpreted as *recall* of changes, which is computed as $H = q_{ch}/Q$, where Q is the total number of queried instances and q_{ch} is the number of queried instances that have their labels changed. In this evaluation we aim to establish a point in time evaluation thus we do not retrain the classifier after each instance. The fixed uncertainty strategy is omitted, because it does not have a mechanism to control the labeling budget. Besides, the fixed uncertainty strategy handles the changes in the same way as the variable uncertainty strategy, only the budget is handled differently.

Comparing the close and the remote change plots we can see, as expected, that the random strategy performs equally well independently of where changes occur. On the other hand, the uncertainty strategy is very sensitive to where changes occur. If changes occur far from the decision boundary, the uncertainty strategy completely misses them and it will never know that the classifier continues making mistakes there. However, the uncertainty strategy captures changes perfectly well if they occur close to the decision boundary, it scores the best of all (100%). The randomized uncertainty reacts to the close changes worse than the uncertainty, but it does not miss the remote changes.

4.2 Learning in Stationary Situations

Active learning strategies in data streams need not only handle changes but also aid the learning to be more accurate. We compare the strategies in terms of queried uncertainty, assuming that the most informative instances in stationary situations lie closest to the decision boundary. Figure 8 (right) plots the queried uncertainty by each strategy against the labeling budget. The plot can

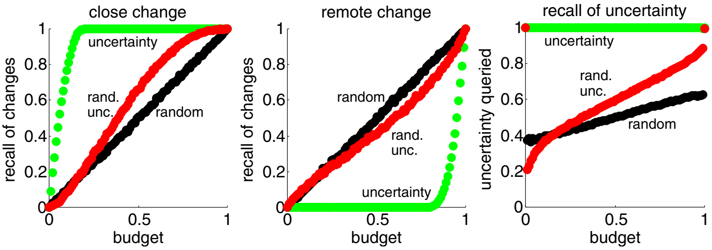


Fig. 8. Performance of the strategies

Table 2. Summary of the datasets

	instances	attributes (nominal + numeric)	classes	labels
Electricity	45312	8 (1+7)	2	original
Cover Type	581012	54 (44+10)	7	original
Airlines	539383	7 (5+2)	2	original
IMDB-E	120919	1000	2	assigned
IMDB-D	120919	1000	2	assigned
Reuters	15564	47236	2	assigned

be interpreted as *recall* of uncertainty, the higher the better. We measure it as $U = 1 - \frac{u_q - \min u}{\max u - \min u}$, where $u_q = \sum_{X queried} \hat{p}(y|X)$ is the sum of the posterior probabilities of all the queried instances, $\min u$ and $\max u$ are the minimum and the maximum possible u_q from our dataset.

The plot confirms that the variable uncertainty always queries the most uncertain instances, thus it is expected to perform well in stationary situations. The random strategy recalls the least, except for very small budgets, where the variable randomized uncertainty strategy recalls even less. This happens because at small budgets the threshold is very small, therefore nearly all randomization attempts override the threshold and query further away from the decision boundary than random. The variable randomized uncertainty strategy becomes more effective as the budget increases. Notice, that the higher the budget, the more similar the performance, since many of the queried instances overlap.

5 Experimental Evaluation

After analyzing our strategies we empirically evaluate their performance along with the baselines. We compare five techniques: random (baseline), fixed uncertainty (baseline), variable uncertainty, variable randomized uncertainty, and *Selective Sampling*. Our implementation of Selective Sampling is based on [5], and uses a variable labeling threshold $b/(b+|p|)$, where b is a parameter and p is the prediction of the classifier. The threshold is based on certainty expectations, the labels are queried at random. As they did not explicitly consider concept drift, we add change detection to the base classifier to improve its performance.

We evaluate the performance on real streaming classification problems. We use as default parameters $s = 0.01$ and $\delta = 1$. All our experiments are performed using the MOA data stream software suite [3]. MOA is an open source software framework in Java designed for on-line settings as data streams. We use in our experiments an evaluation setting based on prequential evaluation: each time we get an instance, first we test it, and if we decide to pay the cost of its label then we use it to train the classifier.

5.1 Datasets

We use six classification datasets as presented in Table 2. Electricity data [9] is a popular benchmark in evaluating streaming classifiers. The task is to predict a rise or a fall in electricity price (demand) in New South Wales (Australia), given recent consumption and prices in the same and neighboring regions. Cover Type data [1] is also often used as a benchmark for evaluating stream classifiers. The task is to predict forest cover type from cartographic variables. As the original coordinates were not available, we ordered the dataset using the elevation feature. Inspired by [12] we constructed an Airlines dataset[2] using the raw data from US flight control. The task is to predict whether a given flight will be delayed, given the information of the scheduled departure.

IMDB data originates from the MEKA repository[3]. The instances are TF-IDF representations of movie annotations. Originally the data had multiple labels that represent categories of movies. We construct binary labels in the following way. At a given time we select categories of interest to an imaginary user, the movies of that category get a positive label. After some time the interest changes. We introduce three changes in the data stream (after 25, 50 and 75 thousand instances). We construct two labelings: for IMDB-E (easy) only one category is interesting at a time; for IMDB-D (difficult) five related categories are interesting at a time, for instance: crime, war, documentary, history and biography are interesting at the same time.

The Reuters data is from [15]. We formed labels from the original categories of the news articles in the following way. In the first half of the data stream legal/judicial is considered to be relevant (+). In the second half the share listings category was considered to be relevant. The categories were selected to make a large enough positive class (nearly 20% of instances had a positive label).

The first three datasets (*prediction datasets*) have original labels. We do expect concept drift, but it is not known with certainty when and if changes take place. The other three datasets (*textual datasets*) represent recommendation tasks with streaming data. Every instance is a document in TF-IDF representation. We form the labels of interest from the categories of the documents.

5.2 Results on Prediction Datasets

We use Naive Bayes as the base classifier for the three prediction datasets. Figure 9 plots the accuracy of a given strategy as a function of the labeling budget. Fixed uncertainty is not included in this figure, since it fails by a large margin. In the data stream scenario it gives around $50\% - 60\%$ accuracy, which is equivalent to predicting that all labels are negative.

Our strategies (variable and randomized uncertainty) outperform the baseline strategies (random in the plots and the fixed uncertainty not in the plots) and selective sampling as follows. We observe that the variable uncertainty strategy is the most accurate on the Airlines data and on a large part of the Electricity data. In stationary situations or when changes happen close to the decision boundary we expect the variable uncertainty to perform the best. For the Electricity data the randomized uncertainty and the selective sampling perform well at small

[2] Our dataset is available at `http://www.cs.waikato.ac.nz/~abifet/active`

[3] http://meka.sourceforge.net/

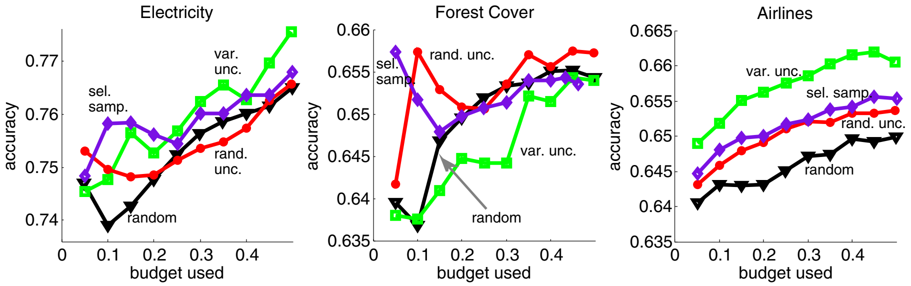


Fig. 9. Accuracies given a budget on prediction datasets

budgets. That is explainable, as at small budgets variable uncertainty samples only a few instances that are very close to the decision boundary. In such a case randomized uncertainty helps to capture changes better. But as soon as the budget increases, variable uncertainty labels more instances and those include the changes. All the plots exhibit rising accuracy as the budget increases, which is to be expected. If there was no upward tendency, then we would conclude that we have excess data and we should be able to achieve a sufficient accuracy by a simple random subsampling.

On the forest cover dataset, randomized uncertainty outperforms the other methods. This learning problem is complex (seven classes), and changes may not happen sufficiently close to the decision boundary, so randomization based methods are best. At small budgets, selective sampling performs well, and when budgets get larger its performance is similar to the random strategy. The variable uncertainty strategy performs well for Airlines, as apparently the data is not changing. The dataset covers only one month, which is a short period for changes to become manifest. Thus randomization of querying strategies does not pay off in this case. Even though more than the minimum number of labels needed, is requested, randomized uncertainty still outperforms the baselines (random and fixed uncertainty). This performance is consistent with our expectations.

5.3 Results on Textual Datasets

For textual datasets we use the Multinomial Naive Bayes classifier [18]. The classification tasks in these textual datasets are hard and often the results are close to the majority vote, which would mean no recommendation is given by a classifier. Therefore we are interested in balanced accuracy of the classifiers, which is given by the geometric mean $GA = (A_1 \times A_2 \times \dots A_c)^{1/c}$, where A_i is the testing accuracy on class i and c is the number of classes. Note that the geometric accuracy of the majority vote classifier would be zero, as accuracy on the classes other than the majority would be zero. The accuracy of a perfectly correct classifier would be one. If the accuracies of a classifier are balanced across the classes, then the geometric accuracy would be equal to the normal accuracy.

Figure 10 presents the geometric accuracies of the three textual datasets. There are several implications following from these results. First, the strategies that use a variable threshold (selective sampling, variable uncertainty, and randomized uncertainty strategy) outperform the fixed threshold strategies (fixed uncertainty), as expected in the data stream setting. Second, the strategies with randomization mostly outperform the strategies without randomization, which suggests that the changes that occur are far from the decision boundaries and there is a justified need for querying tailored to data streams rather than conventional uncertainty sampling. That supports our strategies. Note that as the selective sampling strategy is implemented in our experiments using change detection, it shows good performance on these datasets. However, there is always at least one of the new strategies that outperforms it.

On IMDB-E the random strategy performs the best, while on IMDB-D our randomized uncertainty is the best. This different performance can be explained by the nature of the labels. In IMDB-E one category forms the positive label. These categories do not overlap much, as, for instance, science fiction and sports may have little in common. Thus, the decision boundary changes completely and the change occurs far from the decision boundary. Therefore the random strategy is optimal. That is consistent with our simulation findings. In IMDB-D five categories make a positive label at a time. With a larger space for positive labels it is also likely that there is shared vocabulary in the interests before and after the drift. Thus, the drift happens closer to the decision boundary and thus our randomized uncertainty strategy performs better than the fully random one, which is also in line with our reasoning behind the strategies. The randomized uncertainty strategy performs best on the Reuters data as well, while variable uncertainty comes second. The changes that are happening may not be that far from each other, the concepts before and after the drift may be related.

In the textual datasets we know exactly where the concept drift points are. The progress of the accuracies (prequential) of our strategies and their behavior around the change points Figures 11 and 12. In this active learning experiment we use a fixed 20% budget.

From Figure 11 we can clearly see that when more changes happen, the baseline fixed uncertainty fails to react. The variable uncertainty eventually reacts, but slowly. That demonstrates the need for labeling across all the instance space.

In Figure 12 we closely inspect the behavior with Reuters data. At the start of learning (left) the strategies that use randomization (random and randomized

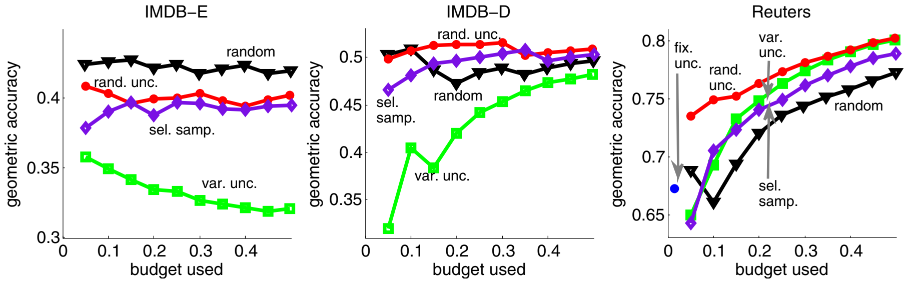


Fig. 10. Accuracies given a budget on textual drifting datasets

uncertainty) learn faster. This happens because simple uncertainty strategies learn a classifier from a few points and this classifier becomes very confident about its own predictions and does not require to learn further. At the stable situation (middle) the strategies without randomization perform better than the strategies with randomization, as expected. There are no changes, thus randomization can be seen as a waste of labeling effort. Fixed uncertainty performs well in stable situations, provided its threshold is set appropriately. At change we see that the strategies with randomization react faster than expected. We also see that the fixed uncertainty strategy fails to adapt. The results at change justifies the need for variable thresholds and randomization of labeling efforts.

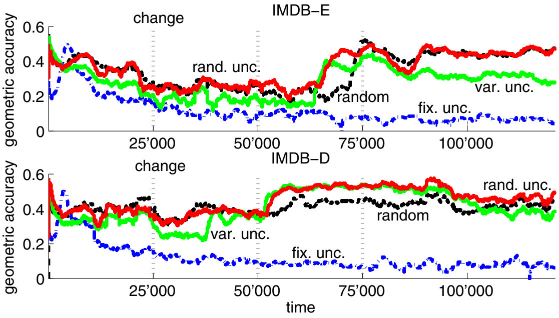


Fig. 11. Progress of accuracies on IMDB drifting datasets

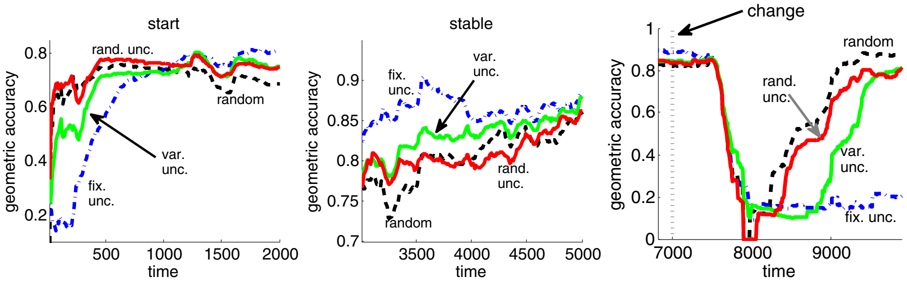


Fig. 12. Progress of accuracies on Reuters dataset

5.4 Efficiency

These active strategies reduce the time and space needed for learning, as the classifiers are trained with a smaller number of instances. We can see these active learning strategies as a way to speed up the training of classifiers: only using 30% or 40 % of the instances we may get only a small decrease on accuracy. For example, in our experiments, labeling all instances ($B = 1$), we see an increase of 5% for the Electricity dataset, 12% for the Cover Type dataset and no increase for the Airlines dataset. On textual data, we obtain an increase of 12% points on the Reuters data set and no change on the IMDB datasets. These results show that these strategies may be a good way to speed up the training process of classifiers.

We introduced new strategies for active learning tailored to data streams when concept drift is expected. Different strategies perform best in different situations. Our recommendation is to use the variable uncertainty strategy if mild to moderate concept drifts are expected. If significant drifts are expected then we recommend using randomized uncertainty. In practice we find that drifts can be captured reliably even though the assumption of i.i.d. data is violated.

6 Conclusion

We proposed active learning strategies for streaming data when changes in the data distribution are expected. Our strategies are equipped with mechanisms to control and distribute the labeling budget over time, to balance the labeling for learning more accurate classifiers and to detect changes.

Experimental results demonstrate that the new techniques are especially effective when the labeling budget is small. The best performing technique depends on what changes are expected. Variable uncertainty performs well in many real cases where the drift is not that strongly expressed. If more significant drift is expected (as in the textual experiments) then the randomized uncertainty prevails, since it is able to query over all the instance space.

This work can be considered as the first step in active learning in the data stream setting. An immediate extension would be to place a grid on the instance space and maintain individual budgets for each region. In such a case it should be possible to dynamically redistribute the labeling budget to the regions where changes are suspected.

Acknowledgements. Part of the research leading to these results has received funding from the EC within the Marie Curie Industry and Academia Partnerships and Pathways (IAPP) programme under grant agreement no. 251617.

References

1. Asuncion, A., Newman, D.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences (2007)
2. Attenberg, J., Provost, F.: Active inference and learning for classifying streams. In: ICML 2010 Workshop on Budgeted Learning (2010)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. Journal of Machine Learning Research 11, 1601–1604 (2010)
4. Borchani, H., Larrañaga, P., Bielza, C.: Mining concept-drifting data streams containing labeled and unlabeled instances. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) IEA/AIE 2010. LNCS, vol. 6096, pp. 531–540. Springer, Heidelberg (2010)
5. Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Worst-case analysis of selective sampling for linear classification. J. Mach. Learn. Res. 7, 1205–1230 (2006)

6. Cohn, D., Atlas, l., Ladner, R.: Improving generalization with active learning. Machine Learning 15, 201–221 (1994)
7. Fan, W., Huang, Y., Wang, H., Yu, P.: Active mining of data streams. In: SDM 2004, pp. 457–461 (2004)
8. 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)
9. Harries, M., Sammut, C., Horn, K.: Extracting hidden context. Machine Learning 32(2), 101–126 (1998)
10. Helmbold, D., Panizza, S.: Some label efficient learning results. In: COLT 1997, pp. 218–230 (1997)
11. Huang, S., Dong, Y.: An active learning system for mining time-changing data streams. Intelligent Data Analysis 11, 401–419 (2007)
12. Ikonomovska, E., Gama, J., Dzeroski, S.: Learning model trees from evolving data streams. Data Mining and Knowledge Discovery 23(1), 128–168 (2010)
13. Klinkenberg, R.: Using labeled and unlabeled data to learn drifting concepts. In: IJCAI Workshop on Learning from Temporal and Spatial Data, pp. 16–24 (2001)
14. Lewis, D., Gale, W.: A sequential algorithm for training text classifiers. In: ACM SIGIR, pp. 3–12 (1994)
15. Lewis, D., Yang, Y., Rose, T., Li, F.: Rcv1: A new benchmark collection for text categorization research. J. Mach. Learn. Res. 5, 361–397 (2004)
16. Lindstrom, P., Delany, S.J., MacNamee, B.: Handling concept drift in a text data stream constrained by high labelling cost. In: FLAIRS. AAAI Press, Menlo Park (2010)
17. Masud, M., Gao, J., Khan, L., Han, J., Thuraisingham, B.: Classification and novel class detection in data streams with active mining. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS, vol. 6119, pp. 311–324. Springer, Heidelberg (2010)
18. Mccallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. In: AAAI Workshop on Learning for Text Categorization (1998)
19. Sculley, D.: Online active learning methods for fast label-efficient spam filtering. In: CEAS 2007 (2007)
20. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
21. Widyantoro, D., Yen, J.: Relevant data expansion for learning concept drift from sparsely labeled data. IEEE Tr. on Know. and Data Eng. 17, 401–412 (2005)
22. Woolam, C., Masud, M., Khan, L.: Lacking labels in the stream: Classifying evolving stream data with few labels. In: Rauch, J., Raś, Z.W., Berka, P., Elomaa, T. (eds.) ISMIS 2009. LNCS, vol. 5722, pp. 552–562. Springer, Heidelberg (2009)
23. Zhang, P., Zhu, X., Guo, L.: Mining data streams with labeled and unlabeled training examples. In: ICDM 2009, pp. 627–636 (2009)
24. Zhu, X., Zhang, P., Lin, X., Shi, Y.: Active learning from data streams. In: ICDM 2007, pp. 757–762 (2007)

Celebrity Watch: Browsing News Content by Exploiting Social Intelligence

Omar Ali, Ilias Flaounas, Tijl De Bie, and Nello Cristianini

Intelligent Systems Laboratory, Bristol University,
Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom
{Omar.Ali,Ilias.Flaounas,Tijl.DeBie,Nello.Cristianini}@bristol.ac.uk
http://patterns.enm.bris.ac.uk

Abstract. Celebrity Watch is an automatically-generated website that presents up-to-date entertainment news from around the world. It demonstrates the application of many pattern analysis methods that allow us to autonomously monitor millions of news articles and hundreds of millions of references to people mentioned in them. We apply statistical methods to merge references into people, track their association to various topics of news, and generate social networks of their co-occurrences in articles. From this sea of data we select the forty most-relevant people and display them on the website, offering users a highly condensed view of the latest in entertainment news. The site updates itself throughout the day and is the final step in a large, fully-autonomous system that monitors online news media.

Keywords: news mining, statistical inference, trend detection, social networks, entertainment news.

1 Introduction

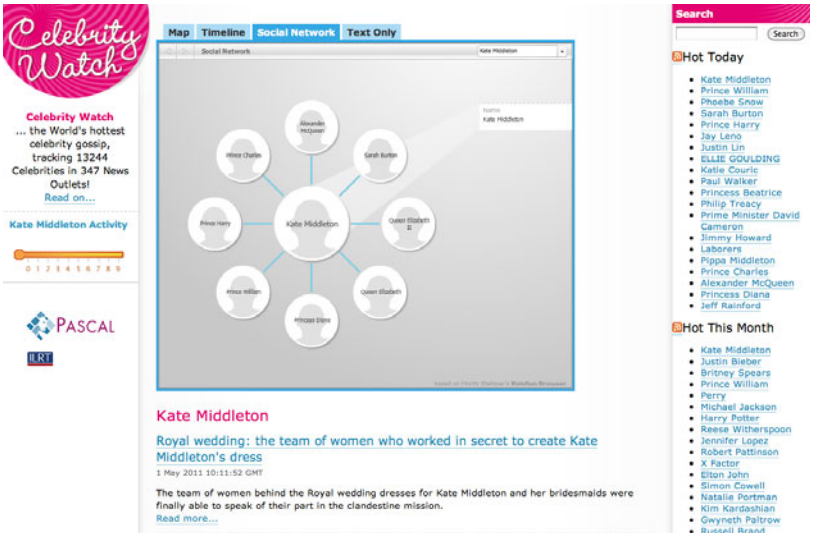
In this paper we present Celebrity Watch, which is a website that delivers the latest entertainment news from the perspective of the people who appear in it. It is an entirely automated system that updates itself many times per day and is able to detect people who are currently 'trending'.[1] Figure 1 shows the website, which is accessible online at http://celebwatch.enm.bris.ac.uk/.

The website has sections dedicated to those people who are trending today, as well as to those who are trending this month. In the first case we see people who are mentioned in breaking entertainment news, while in the second we see the most popular celebrities of the moment.

Associated with each person are a selection of timeline views depicting their media activity in recent history. Each person also has an automatically inferred social network, which allows users to browse those who are most connected to them, as seen in online news.

[1] The term 'trending' is taken from Twitter (http://www.twitter.com), and refers to a sudden increase in usage of a term on their website.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 613–616, 2011.



Celebrity Watch
... the World's hottest celebrity gossip, tracking 13244 Celebrities in 347 News Outlets!
Read on...

Kate Middleton

Royal wedding: the team of women who worked in secret to create Kate Middleton's dress

1 May 2011 10:11:52 GMT

The team of women behind the Royal wedding dresses for Kate Middleton and her bridesmaids were finally able to speak of their part in the clandestine mission.
Read more...

Hot Today

- Kate Middleton
- Prince William
- Phoebe Snow
- Sarah Burton
- Prince Harry
- Jay Leno
- Justin Lin
- ELLIE GOULDING
- Katie Couric
- Paul Walker
- Princess Beatrice
- Philip Treacy
- Prime Minister David Cameron
- Jimmy Howard
- Laborers
- Pippa Middleton
- Prince Charles
- Alexander McQueen
- Princess Diana
- Jeff Rainford

Hot This Month

- Kate Middleton
- Justin Bieber
- Britney Spears
- Prince William
- Perry
- Michael Jackson
- Harry Potter
- Reese Witherspoon
- Jennifer Lopez
- Robert Pattinson
- X Factor
- Elton John
- Simon Cowell
- Natalie Portman
- Kim Kardashian
- Gwyneth Paltrow
- Russell Brand

Fig. 1. An image of the main page of Celebrity Watch, centred on Kate Middleton's social network (taken on 2nd May 2011)

Celebrity Watch is the result of an autonomous software pipeline that collects articles from over one thousand online news outlets. The pipeline extracts and resolves references to people mentioned in these articles using statistical methods and multiple sources of information. Our system currently monitors over 15 million English-language articles and tracks over 36 thousand people. A key challenge of building such a system is extracting a meaningful signal of sufficient quality from a large volume of data, while allowing fast and easy access to the latest changes in the world.

News articles are collected for a whole range of topics, and their statistical association to each person is automatically monitored. This allows us to detect interesting people—those whose association to entertainment news is suddenly increasing—and report them to the world immediately. Note that 'entertainment' is just a parameter in the construction of this website; in actual fact we could change this to 'sport' or 'business' to generate a similar site with a different focus.

We should reiterate that the entire pipeline is automated. It collects and labels articles by their topic, extracts and resolves references to people, monitors their association to entertainment news, detects interesting changes in their association, generates social networks, and updates itself throughout the day. All of this without human intervention.

This paper briefly outlines the components of our software pipeline that contribute to the operation of Celebrity Watch. It is updated at regular intervals every day and can be accessed online at http://celebwatch.enm.bris.ac.uk/.

2 Methods

Celebrity Watch is generated as a result of many software modules that interact via a number of databases. Online news websites are monitored by a multi-threaded spider, which downloads and parses news feeds in order to populate a database of articles. Much of our data-collection system is already described elsewhere [5], so we omit full details here.

References to named entities are extracted using GATE [3], prior to applying large-scale entity matching using multiple sources of information [1].

The result of these steps is a set of over 36 thousand people linked to over 15 million articles, all of which is automatically processed. In the following section we outline how we infer the topic of a person and how we track their association to entertainment news such that we can immediately detect changes in it. We then explain how social networks are generated, before briefly explaining how we rank articles and geo-locate them.

Topic Tracking: We track the statistical association between named entities the topics of news with which they are associated. This is measured using the odds ratio [2]. All entity–topic associations are tracked using a set of exponentially weighted moving averages. Each of these decays with half-lives of one day, one week, one month, or one year; the fast-decaying averages reflect media activity of a person at the present moment, while the slow-decay ones reflect long-term trends in any media topic. Large increases in association of a person to any topic is typically evidence of a new story about them. We detect such increases, or trends, by *comparing* moving averages.

Celebrity Watch is generated many times each day and its focus is dedicated to only 40 people: the top 20 movers today as compared to this week, and the top 20 movers this month as compared to this year. Our infrastructure allows us to generate a list of the biggest movers within any topic of news, within a few seconds. The remaining steps collect other information from our systems to bring this information together to form Celebrity Watch.

Social Networks: Social networks are generated based on co-occurrences between people seen in entertainment articles. We first construct a master network that considers co-occurrences between people mentioned in entertainment articles seen in the past 30 days. This network is filtered using the χ^2 test of independence, so that only the most significant connections are maintained. This step takes under 10 minutes and is run once per day.

We further filter this network to include only the 40 people of interest at present, as well as those who are within two steps of them in the network. This step takes a few seconds and ensures that the site remains focused on the most interesting people of the moment.

Assembly and Presentation: Final steps in the production of Celebrity Watch include the addition of the most recent articles that mention each person. These are ranked according to where in the article a person is mentioned. Only the title and summary of each article is displayed, along with a link to the source web page. In addition, we use extracted locations from the text of each article to geo-locate them, to display on the site map.

3 Discussion and Conclusions

The barrage of information delivered on the web is unlikely to slow down, so we need ways to reduce this data overload and focus our attention on that which is most interesting to us. In this case we have presented a digest of the most interesting news by presenting a handful of people of interest, discovered among millions of possibilities.

It demonstrates what can be achieved by integrating multiple machine learning and text mining technologies into a unified, autonomous system. Our approach contrasts with existing systems, [4], [6], in that it presents news stories by detecting the *people* who are most interesting in the world at present.

Further steps could leverage our natural tendency to process emotions associated to news and social relations. We intend to add to the social aspect of the site, so that users may browse the news from a social perspective, and better map to our socially-orientated brains.

Acknowledgements. We would like to thank Simon Price and Ben Joyner of the University of Bristol Institute for Learning and Research Technology, for their work on the web site front-end.

References

1. Ali, O., Cristianini, N.: Information Fusion for Entity Matching in Unstructured Data. In: Papadopoulos, H., Andreou, A.S., Bramer, M. (eds.) AIAI 2010. IFIP Advances in Information and Communication Technology, vol. 339, pp. 162–169. Springer, Heidelberg (2010)
2. Boslaugh, S., Watters, P.A.: Statistics in a Nutshell: A Desktop Quick Reference (In a Nutshell (O'Reilly)). O'Reilly Media, Sebastopol (2008)
3. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A framework and graphical development environment for robust NLP tools and applications. In: Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics, pp. 168–175 (2002)
4. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, ACM, New York (2007)
5. Flaounas, I., Ali, O., Turchi, M., Snowsill, T., Nicart, F., De Bie, T., Cristianini, N.: NOAM: News Outlets Analysis and Monitoring System. In: SIGMOD (2011), accepted for publication
6. Pouliquen, B., Steinberger, R., Deguernel, O.: Story tracking: linking similar news over time and across languages. In: Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization, MMIES 2008, pp. 49–56. Association for Computational Linguistics, Stroudsburg (2008)

MOA: A Real-Time Analytics Open Source Framework

Albert Bifet[1], Geoff Holmes[1], Bernhard Pfahringer[1], Jesse Read[1],
Philipp Kranen[2], Hardy Kremer[2], Timm Jansen[2], and Thomas Seidl[2]

[1] Department of Computer Science, University of Waikato, Hamilton, New Zealand
{abifet,geoff,bernhard,jmr30}@cs.waikato.ac.nz
[2] Data Management and Exploration Group, RWTH Aachen University, Germany
{kranen,kremer,jansen,seidl}@cs.rwth-aachen.de

Abstract. **M**assive **O**nline **A**nalysis (MOA) is a software environment for implementing algorithms and running experiments for online learning from evolving data streams. MOA is designed to deal with the challenging problems of scaling up the implementation of state of the art algorithms to real world dataset sizes and of making algorithms comparable in benchmark streaming settings. It contains a collection of offline and online algorithms for classification, clustering and graph mining as well as tools for evaluation. For researchers the framework yields insights into advantages and disadvantages of different approaches and allows for the creation of benchmark streaming data sets through stored, shared and repeatable settings for the data feeds. Practitioners can use the framework to easily compare algorithms and apply them to real world data sets and settings. MOA supports bi-directional interaction with WEKA, the Waikato Environment for Knowledge Analysis. Besides providing algorithms and measures for evaluation and comparison, MOA is easily extensible with new contributions and allows for the creation of benchmark scenarios.

1 Introduction

In data stream scenarios data arrives at high speed strictly constraining processing algorithms in space and time. To adhere to these constraints, specific requirements have to be fulfilled by the stream processing algorithms, that are different from traditional batch processing settings. The most significant requirements are the following: process an example at a time, and inspect it at most once; use a limited amount of memory; work in a limited amount of time; and be ready to predict at any time.

Stream learning algorithms are an important type of stream processing algorithm: in a repeated cycle, the learned model is constantly updated to reflect the incoming examples from the stream. They do so without exceeding their memory and time bounds. After processing an incoming example, the algorithms are always able to output a model. Typical learning tasks in stream scenarios are classification, regression, clustering, and frequent pattern mining.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 617–620, 2011.

MOA is an open-source framework for dealing with massive evolving data streams. It is the first data mining software designed specifically for data streams to include multi-label classification and graph mining methods, in addition to regular classification and clustering methods [3].

Our stream learning framework provides a set of data generators, algorithms and evaluation measures. Practitioners can benefit from this by comparing several algorithms in real world scenarios and choosing the best solution. For researchers our framework yields insights into advantages and disadvantages of different approaches and allows the the creation of benchmark streaming data sets through stored, shared and repeatable settings for the data feeds. The sources are publicly available and are released under the GNU GPL license.

Only two other open-source data streaming packages exist: VFML and a RapidMiner plugin. The VFML (Very Fast Machine Learning) [4] toolkit was the first open-source framework for mining high-speed data streams and very large data sets. It was developed until 2003. VFML is written mainly in standard C, and contains tools for learning decision trees (VFDT and CVFDT), for learning Bayesian networks, and for clustering.

The data stream plugin (formerly: concept drift plugin) [5] for RapidMiner (formerly: YALE (Yet Another Learning Environment)), is an extension to RapidMiner implementing operators for handling real and simulated concept drift in evolving streams.

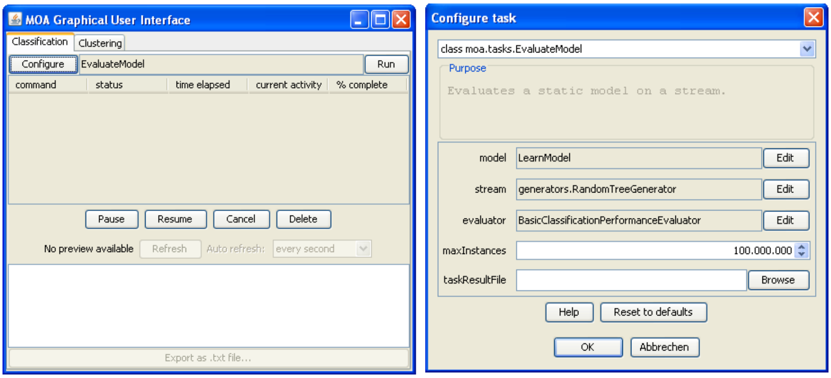
MOA is built on experience with both WEKA and VFML. The main advantage of MOA is that it provides many of the recently developed data stream algorithms, including learners for multi-label classification and graph mining. It also contains a graphical interface, and the software is built using object-oriented techniques. Generally, it is straightforward to use or to extend MOA.

2 Experimental Framework

MOA is written in Java. The main benefits of Java are portability, where applications can be run on any platform with an appropriate Java virtual machine, and the strong and well-developed support libraries. Use of the language is widespread, and features such as automatic garbage collection help to reduce programmer burden and error.

MOA contains stream generators, learners and evaluation methods. Figure 1 shows the MOA graphical user interface. However, a command line interface is also available. Considering data streams as data generated from pure distributions, MOA models a concept drift event as a weighted combination, or mixture distribution, of two pure distributions that characterize the target concept before and after the drift. The mixing proportion for the “after” concept smoothly increases from zero to one inside a user-defined window around the time point of change. The increase follows a sigmoid function, an elegant and practical solution [2].

MOA streams can be built using generators, reading ARFF files, joining several streams, or filtering streams. Most of the data generators commonly found



EvaluateModel

class moa.tasks.EvaluateModel

Evaluates a static model on a stream.

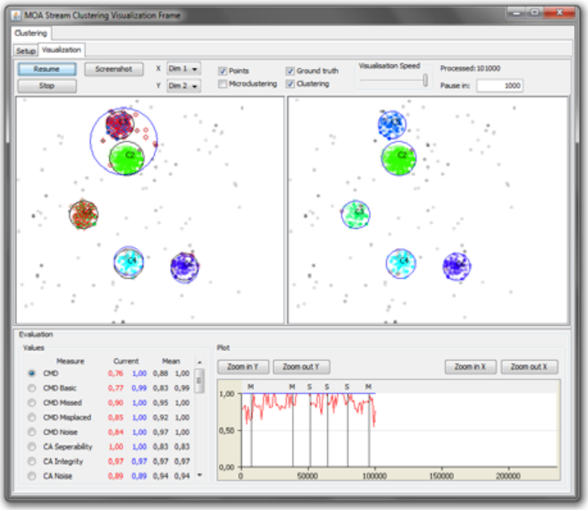
model	LearnModel
stream	generators.RandomTreeGenerator
evaluator	BasicClassificationPerformanceEvaluator
maxInstances	100.000.000
taskResultFile	

Fig. 1. MOA Graphical User Interface

in the literature, are provided: Random Tree Generator, SEA Concepts Generator, STAGGER Concepts Generator, Rotating Hyperplane, Random RBF Generator, LED Generator, Waveform Generator, and Function Generator.

MOA contains a range of classification methods such as: Naive Bayes, Stochastic Gradient Descent, Perceptron, Hoeffding Tree, Adaptive Hoeffding Tree, Hoeffding Option Tree, Bagging, Boosting, Bagging, and Leveraging Bagging.

For clustering [6], MOA contains several stream clustering methods such as StreamKM++, CluStream, ClusTree, Den-Stream, and CobWeb. Dynamic visualization of cluster evolution is available, as depicted in Figure 2.



☑ Points
☐ Microclustering
☑ Ground truth
☑ Clustering

Processed: 101000

Pause in: 1000

	Measure	Current		Mean	
●	CMD	0,76	1,00	0,88	1,00
○	CMD Basic	0,77	0,99	0,83	0,99
○	CMD Missed	0,90	1,00	0,95	1,00
○	CMD Misplaced	0,85	1,00	0,92	1,00
○	CMD Noise	0,84	1,00	0,97	1,00
○	CA Seperability	1,00	1,00	0,83	0,83
○	CA Integrity	0,97	0,97	0,97	0,97
○	CA Noise	0,89	0,89	0,94	0,94

Fig. 2. Visualization tab of the clustering MOA graphical user interface

Two recent extensions to MOA are multi-label classification and graph mining. In multi-label classification, instead of a single class-label, each example can be associated with *multiple* labels. Multi-label classification has seen considerable development in recent years, but so far most of this work has been carried out in the context of batch learning where train-then-test or cross-fold validation evaluations are typical. MOA implements multi-label stream generators and several state of the art methods: ECC Ensembles of classifier-chains, and EPS Ensembles of Pruning Sets, Multi-label Hoeffding Trees, and multi-label adaptive bagging methods.

MOA also contains a framework for studying graph pattern mining on time-varying streams [1]. All methods work on coresets of closed subgraphs, compressed representations of graph sets, and maintain these sets in a batch-incremental manner, but use different approaches to address potential concept drift. MOA implements IncGraphMiner, WinGraphMiner and AdaGraphMiner.

2.1 Website, Tutorials, and Documentation

MOA can be found at: `http://moa.cs.waikato.ac.nz/` The website includes a tutorial, an API reference, a user manual, and a manual about mining data streams. Several examples of how the software can be used are available. We are currently working on extending the framework to include data stream regression, and frequent pattern learning.

Acknowledgments. This work has been supported by the UMIC Research Centre, RWTH Aachen University, Germany.

References

1. Bifet, A., Holmes, G., Pfahringer, B., Gavaldà, R.: Mining frequent closed graphs on evolving data streams. In: 17th ACM SIGKDD (2011)
2. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: 15th ACM SIGKDD (2009)
3. Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., Seidl, T.: Moa: Massive online analysis, a framework for stream classification and clustering. Journal of Machine Learning Research - Proceedings Track 11, 44–50 (2010)
4. Hulten, G., Domingos, P.: VFML – a toolkit for mining high-speed time-changing data streams (2003)
5. Klinkenberg, R.: Rapidminer data stream plugin. RapidMiner (2010), `http://www-ai.cs.uni-dortmund.de/auto?self=$eit184kc`
6. Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., Pfahringer, B.: An effective evaluation measure for clustering on evolving data stream. In: 17th ACM SIGKDD (2011)

L-SME: A System for Mining Loosely Structured Motifs

Fabio Fassetti [1], Gianluigi Greco [2], and Giorgio Terracina [2]

[1] ICAR-CNR
[2] Dep. of Mathematics, Via P. Bucci, 87036 Rende (CS), Italy
ffassetti@deis.unical.it, {ggreco,terracina}@mat.unical.it

Abstract. We present L-SME, a system to efficiently identify loosely structured motifs in genome-wide applications. L-SME is innovative in three aspects. Firstly, it handles wider classes of motifs than earlier motif discovery systems, by supporting boxes swaps and skips in the motifs structure as well as various kinds of similarity functions. Secondly, in addition to the standard exact search, it supports search via randomization in which guarantees on the quality of the results can be given a-priori based on user-definable resource (time and space) constraints. Finally, L-SME comes equipped with an intuitive graphical interface through which the structure for the motifs of interest can be defined, the discovery method can be selected, and results can be visualized. The tool is flexible and scalable, by allowing genome-wide searches for very complex motifs and is freely accessible at http://siloe.deis.unical.it/l-sme. A detailed description of the algorithms underlying L-SME is available in [1].

1 Introduction

Transcriptional control is a crucial mechanism for gene regulation, in which certain proteins, called transcription factors, bind near genes to activate or inhibit the transcription of genetic information from DNA to RNA. Transcription factors are known to have special affinity for short DNA regions called binding sites, which occur several times in the same genome and which are conserved in evolution over different organisms [5]. Thus, singling out the regions that are over-represented in suitably selected sets of DNA sequences provides us with insights on the biological functions played by the corresponding macromolecules [3]. These regions are called *motifs* in the literature.

Several discovery methods and tools have already been conceived to identify motifs conforming to some model templates that capture the similarities of diverse binding sites, and which are fixed by the biologist who is willing to corroborate his hypothesis on the co-regulation of some given genes. In its basic form, a model template is just the specification of a length l for sequences of DNA basis (called *boxes*); thus, a motif conforming to such a template is precisely a sequence of l basis that is frequently repeated over the genome at hand. More complex model templates, instead, are supported in a few state-of-the-art systems—see, e.g., the comparative analysis by [1]—in order to look for motifs *(i)* that are made of several boxes at a given distance over the gene (called *gap*) from one another [2,6,3], and *(ii)* that may be partially conserved in the repetitions, since *mismatches* are allowed.

In this paper, the L-SME motif discovery tool is presented. In addition to supporting classical model templates, the tool allows to specify box swaps and skips as well

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 621–625, 2011.

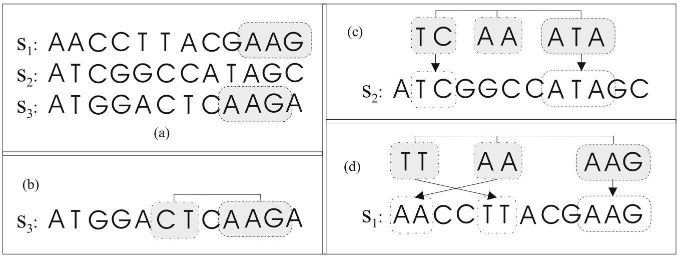


Fig. 1. Illustration of model templates and instances

as various similarity functions to handle mismatches, all of them being variabilities of interest in the context of analyzing eukaryotic transcription [4,7]. In order to handle wider classes of model templates than existing systems while still guaranteeing scalability over genome-wide applications, L-SME is founded on specialized data-structures and advanced computational methods supporting both exact and randomized searches.

2 Motif Discovery Problem

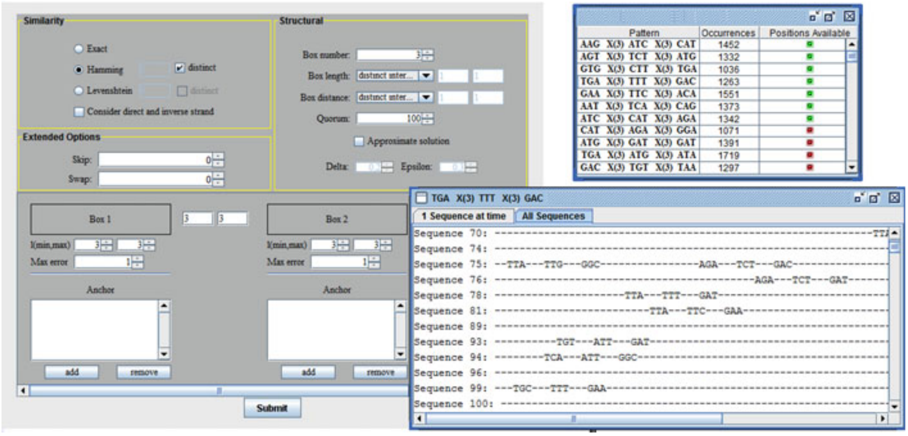
In state-of-the-art discovery tools, a motif template $\widehat{p}$ can be viewed as a tuple $\langle l_1, d_1, l_2, d_2, ..., d_{r-1}, l_r \rangle$, where l_i indicates the length of the i-th box, d_j indicates the length of the gap separating the j-th and the $(j+1)$-th boxes, and r is the number of boxes—actually, both l_i and d_j can be (possibly degenerating) intervals of the form $l_i = [\min\_l_i : \max\_l_i]$ and $d_j = [\min\_d_j : \max\_d_j]$. A *pattern instance* p for $\widehat{p}$ is a string $p = b_{l_1}\ X(d_1)\ b_{l_2}\ X(d_2) ... X(d_{r-1})\ b_{l_r}$, where b_{l_i} are strings, the length of b_{l_i} is in the range $[\min\_l_i : \max\_l_i]$, and $X(d_j)$ is a sequence of d_j special ("don't care") symbols X with length in the interval $[\min\_d_j : \max\_d_j]$. We say that the instance p *occurs* in a DNA sequence s if there is a substring s' of s that matches with p, i.e., such that the *Hamming* or *Levenshtein distance* between each box in p and the corresponding sequence of symbols in s' is below a given threshold (denoting the number of allowed mismatches). Eventually, the *motif discovery problem* over a set of DNA sequences is to find all the instances for $\widehat{p}$ that occur in at least Q of them, where Q is the *quorum* considered appropriate by the biologist for the application at hand.

As an example, over the sequences depicted in Figure 1(a), AAG is the only solution for Q=2, for a model template made by one box of exactly three symbols, and when no mismatches are allowed. As a further example, a more complex template composed of two boxes separated by one irrelevant symbol is shown in Figure 1(b) together with an instance for it occurring in s_3.

3 System Functionalities

L-SME is a tool for motif discovery supporting various innovative functionalities, under various different perspectives.

(1) Supported-Templates Perspective: The tool deals with a wider class of model templates than those discussed in Section 2. Indeed, in addition to those elements, L-SME



Similarity

○ Exact
● Hamming ☑ distinct
○ Levenshtein □ distinct
□ Consider direct and inverse strand

Extended Options

Skip: 0
Swap: 0

Structural

Box number: 3
Box length: distinct inter...
Box distance: distinct inter...
Quorum: 100
□ Approximate solution
Delta: 0 Epsilon: 0

Box 1: 3 3
l(min,max) 3 3
Max error 1

Box 2
l(min,max) 3 3
Max error 1

Pattern	Occurrences	Positions Available
AAG X(3) ATC X(3) CAT	1452	🟩
AGT X(3) TCT X(3) ATG	1332	🟩
GTG X(3) CTT X(3) TGA	1036	🟩
TGA X(3) TTT X(3) GAC	1263	🟩
GAA X(3) TTC X(3) ACA	1551	🟩
AAT X(3) TCA X(3) CAG	1373	🟩
ATC X(3) CAT X(3) AGA	1342	🟩
CAT X(3) AGA X(3) GGA	1071	🟥
ATG X(3) GAT X(3) GAT	1391	🟥
TGA X(3) ATG X(3) ATA	1719	🟥
GAC X(3) TGT X(3) TAA	1297	🟥

TGA X(3) TTT X(3) GAC

```
Sequence 70: ----------------------------------------------------------TTA
Sequence 74: ---------------------------------------------------------------
Sequence 75: --TTA---TTG---GGC---------------AGA---TCT---GAC---------------
Sequence 76: ------------------------------------------AGA---TCT---GAT------
Sequence 78: ---------------------TTA---TTT---GAT---------------------------
Sequence 81: -------------------------TTA---TTC---GAA----------------------
Sequence 89: ---------------------------------------------------------------
Sequence 93: ----------TGT---ATT---GAT-------------------------------------
Sequence 94: --------TCA---ATT---GGC---------------------------------------
Sequence 96: ---------------------------------------------------------------
Sequence 99: ---TGC---TTT---GAA---------------------------------------------
Sequence 100: --------------------------------------------------------------
```

Fig. 2. Screenshots of L-SME

supports two further variabilities clearly emerged from recent studies [4,7]. Specifically, L-SME allows patterns to be matched with some given strings even though:

- [*Box skips*] up to a certain user-definable number of boxes is not preserved at all. E.g., Figure 1(c) shows a pattern instance with three boxes matching with s_2 provided one box skip is allowed.
- [*Box swaps*] the relative positions of two consecutive boxes is inverted, where users may specify the maximum number of allowed inversions. E.g., Figure 1(d) shows a pattern instance matching with s_1 provided one box swap.

Moreover, differently from current systems, which are designed to deal with one fixed similarity function only (either Hamming or Levenshtein distance), the similarity function to be used with L-SME can be freely selected by the biologist.

(2) Algorithmic Perspective. Given the need to handle wide classes of templates while guaranteeing scalability over genome-wide applications, L-SME supports search via randomization, in which a-priori guarantees on the quality of the results can be given based on user-definable resource (time and space) constraints. Randomization is based on using *sketches* to store pattern occurrences. Specifically, users can trim two normalized coefficients δ and ϵ to indicate the amount of space to be used for each sketch and the range of tolerance admitted over the accuracy of the solutions, respectively. Higher values reduce time/space requirements but also results quality guarantees. For $\delta = \epsilon = 0$, the randomized approach degenerates to the exact search.

(3) Interfacing Perspective. Given the wide range of parameters handled by L-SME, its user interface is carefully designed so as to simplify the setting-up phases of biological experimentations. In fact, differently from most of the other tools in the literature, L-SME is equipped with a web-based interface where both the process of specifying the model template with the parameters of interest, and the navigation of results can be carried out in a visual and interactive manner. In particular, for each motif that is discovered, L-SME allows the user to visualize its occurrences over the input sequences, which is often very helpful for the biologist.

Table 1. Binding sites information for UASH and URS1H in Saccharomyces cerevisiae. Here ORF stands for Open Reading Frame.

Gene #	ORF	Gene ID	Mapped Site for UASH	Gap	Mapped Site for URS1H
1	YDR285W	ZIP1	GATTCGGAAGTAAAA	5	TCGGCGGCTAAAT
2	YER044C-A	MEI4	TCTTTCGGAGTCATA	8	TGGGCGGCTAAAT
3	YER179W	DMC1	TTGTGTGGAGAGATA	17	AAATAGCCGCCCA
4	YHR014W	SPO13	TAATTAGGAGTATAT	4	AAATAGCCGCCGA
5	YNL210W	MER1	GGTTTTGTAGTTCTA	22	TTTTAGCCGCCGA
6	YHR153C	SPO16	CATTGTGATGTATTT	96	TGGGCGGCTAAAA
7	YHR157W	REC104	CAATTTGGAGTAGGC	74	TTGGCGGCTATTT
8	YLR263W	RED1	ATTTCTGGAGATATC	173	TCAGCGGCTAAAT
9	YMR133W	REC114	GATTTTGTAGGAATA	179	TGGGCGGCTAACT
10	YOR351C	MEK1	TCATTTGTAGTTTAT	179	ATGGCGGCTAAAT
11	YIL072W	HOP1	TGTGAAGT	-323	ATGGCGGCTAAAT

(4) Computation Perspective. Finally, since motif discovery is a computationally intensive task, L-SME is designed to incrementally produce results. In fact, each request is immediately answered with an *url* where discovered results are visualized as soon as they are discovered by internal algorithms, and remain available for some days.

Example Usage. As an example of the results that can be obtained with our system, coupled with a Z-score analysis, we consider here the Saccharomyces cerevisiae, for which several (single) transcription factors are well-known, with some of them being recognized to cooperatively regulate the corresponding genes. For instance, the transcription factors URS1H and UASH are involved in early meiotic expression during sporulation and are known to cooperate for the expression of 11 genes. Table 1 summarizes the genes involved, the transcription factors and the relative positions, which were annotated by biologists and confirmed by our system. The negative gap reported for HOP1 indicates that, in this gene, the relative positions of the two factors are actually swapped w.r.t. all the other genes; this occurrence would not be derived without the support of box swaps and, hence, in current systems available in the literature.

References

1. Fassetti, F., Greco, G., Terracina, G.: Mining loosely structured motifs from biological data. IEEE Transaction on Knowledge and Data Engineering 20(11), 1472–1489 (2008)
2. Hughes, J.D., Estep, P.W., Tavazoie, S., Church, G.M.: Computational identification of cis-regulatory elements associated with groups of functionally related genes in saccharomyces cerevisiae. Journal of Molecular Biology 296(5), 1205–1214 (2000)
3. Marsan, L., Sagot, M.-F.: Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. Journal of Computational Biology 7(3-4), 345–362 (2000)
4. Osanai, M., Takahashi, H., Kojima, K.K., Hamada, M., Fujiwara, H.: Essential motifs in the 3' untranslated region required for retrotransposition and the precise start of reverse transcription in non-long-terminal-repeat retrotransposon SART1. Mol. Cell. Biol. 24(19), 7902–7913 (2004)
5. Sandve, G.K., Drabls, F.: A survey of motif discovery methods in an integrated framework. Biology Direct 1(11), 1–16 (2006)

6. Sinha, S., Tompa, M.: YMF: A program for discovery of novel transcription factor binding sites by statistical overrepresentation. Nucleic Acid Research 31(13), 3586–3588 (2003)
7. Tu, Z., Li, S., Mao, C.: The changing tails of a novel short interspersed element in aedes aegypti: genomic evidence for slippage retrotransposition and the relationship between 3' tandem repeats and the poly(da) tail. Genetics 168(4), 2037–2047 (2004)

The MASH Project

François Fleuret, Philip Abbet, Charles Dubout, and Leonidas Lefakis

Idiap Research Institute, Martigny, Switzerland
{francois.fleuret,philip.abbet,charles.dubout,leonidas.lefakis}@idiap.ch

Abstract. It has been demonstrated repeatedly that combining multiple types of image features improves the performance of learning-based classification and regression. However, no tools exist to facilitate the creation of large pools of feature extractors by extended teams of contributors.

The MASH project aims at creating such tools. It is organized around the development of a collaborative web platform where participants can contribute feature extractors, browse a repository of existing ones, run image classification and goal-planning experiments, and participate in public large-scale experiments and contests.

The tools provided on the platform facilitate the analysis of experimental results. In particular, they rank the feature extractors according to their efficiency, and help to identify the failure mode of the prediction system.

Keywords: pattern recognition, image features, collaborative design.

1 Introduction

Research in Artificial Intelligence has historically focused on two diverse approaches each with their own advantages: Symbolic methods which allow the hand-design of rich prior knowledge under the form of large sets of formal rules, and statistical methods which can cope with the unpredictability and the randomness of real-world situations.

At a crossroads between these two approaches, it has systematically been shown that increasing the complexity of learning systems, for instance by combining algorithms developed independently by different groups of experts is a successful strategy. The Netflix challenge was ultimately won by combining multiple predictors, developed by different teams [1]. In object recognition, state-of-the-art performances are attained by combining multiple feature extractors, with simple learning techniques, e.g. [2]. Our own experiments on the INRIA pedestrian data set show that using a Boosting learning scheme in conjunction with multiple feature extractors, as opposed to solely the best, leads to a reduction in error rate from $\simeq 1\%$ to $\simeq 0.3\%$.

Despite this evidence and the admitted long-term goal of machine learning to produce systems dealing with versatile real-world challenges, no tools have been invented for the design of complex learning architectures by extended teams of people. Though tools such as version control systems or general modeling

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 626–629, 2011.

methods facilitate some classical aspects of development, for which modular and deterministic specifications can be drafted, the heart of a learning system requires a statistical and redundant approach. While it may be possible to predict from the formal definition of a learning method some crude aspects of its behavior, such as convergence or robustness to over-fitting, it is extremely difficult to foresee its performance on real data. This fog of machine learning requires a systematic experimental evaluation to an extent which is not handled properly by existing software development tools and methodologies.

2 The MASH Platform

In order to address this structural complexity, we have developed a framework, centered around a web platform, to study the collaborative design of very large families of feature extractors (see figures 1 and 2).

We call "heuristic" an algorithm that computes a feature vector at any scale and location in an image, and possesses a persistent state (see figure 3). This definition is general enough to allow classical pre-processing from the computer vision world, such as edge detectors, color histograms, SIFT, HOG, LBP, etc. and leads to a clear and simple specification as a C++ class implementing a few methods. The persistent state is irrelevant to image classification, but is used for goal-planning with POMDP, one of the target applications of the project.

Participants to the project may download a multi-platform Software Development Kit (SDK) which allows them to develop and test MASH heuristics on their Linux, Microsoft Windows, or Mac OSX machine. The platform gives users access to documentation, including screen-cast tutorials, regarding the functionality of the web platform itself as well as the development of the heuristics. The platform is further equipped with a forum and private messaging which allow user interaction and communication.

Registered users are able to upload heuristics to the platform which are then stored in those users' "private spaces" and are accessible only by them. The platform allows these contributors to run experiments which provide an evaluation

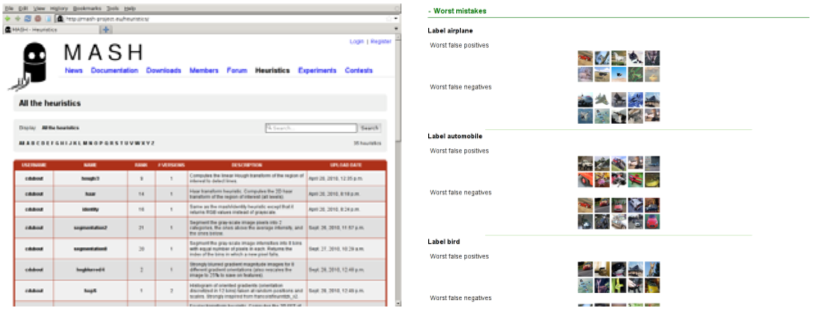


Fig. 1. The MASH platform at `http://mash-project.eu`

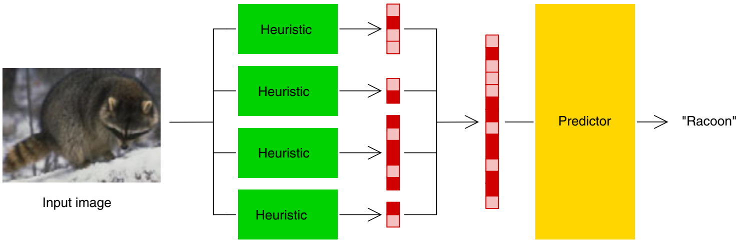


Fig. 2. Feature extraction and processing of the resulting signal

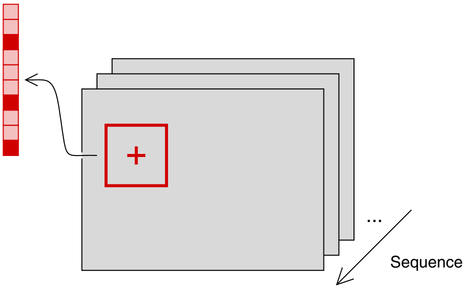


Fig. 3. A heuristic is an image feature extractor with a persistent state

of the quality of a contribution without public exposure; these experiments are run remotely on the platform's servers, thus freeing the users' personal computational resources.

A contributor can at any moment move heuristics to the public space, in which case they become visible, are included in the large-scale public experiments which aim to develop complex state-of-the-art systems, and can also be re-used by any other contributor under an open-source license. Feature extractors already public include well-known methods (e.g. HoG, LBP), as well as novel methods developed on the framework. The source code of all public heuristics is available under the GPL v2 license.

The system tests performances on three families of applications : image classification, object detection, and goal-planning in a simulated environment (reminiscent of modern 3D video-games) and with a real robotic arm. Experiments are run concurrently on multiple machines and the platform aggregates the results in a synthetic manner. It highlights the strengths and weaknesses of the up-to-date resulting trained predictors, and returns to the user a number of quantitative and qualitative evaluations of the results, such as the raw accuracy of a trained classifier, or samples of the worst mistakes on the test data-set (see Fig. 1, right). The standard interaction between a contributor and the platform is to alternate between private experiments to try "new ideas", and public experiments in which their heuristics aid machine learning modules in their identified shortcomings.

3 Contests

In addition to the usual tasks the platform addresses, the web platform also periodically holds contests where registered users can compete on a specific task and data-set.

At the moment, the website is running a three-track contest on image classification based on the CIFAR data-set [3]. We have for each track trained with Boosting a strong predictor composed of N stumps – the tracks corresponding to $N = 0$, $N = 100$ and $N = 10,000$ respectively – using all the heuristics we have already implemented. For each new heuristic participating in the contest, the system runs 100 additional iterations of Boosting, and computes the gain in test error, which is used as the performance measure for that contesting heuristic.

The best heuristic in each track will be selected every month, and added to the pool of heuristics used to train the strong classifiers, which will be subsequently re-trained. Beyond competing with fellow users and aiming to win a prize which will be offered to the "best" feature extractor, users have the opportunity to contribute to the ongoing development of a sophisticated state-of-the-art classification system. The system's current trained classifier for the CIFAR data-set already attains state-of-the-art performance and aims to surpass this with the contributors' help.

4 Conclusion

We advocate the need for a new research domain investigating complex learning systems. In order to develop a unified and centralized artificial intelligence, able to deal with the real world's versatility and complexity, we will first have to develop the requisite, and novel, tools for hand-designing such architectures by hundreds or even thousands of contributors.

The MASH platform is a first initiative in this direction. It targets architectures combining large sets of feature extractors with standard learning procedures. It allows multiple contributors to combine their efforts, it hosts multiple algorithms and runs multiple experiments transparently to assess performance continuously on multiple tasks.

Acknowledgments. This work was supported by the European Community's 7th Framework Programme under grant agreement 247022 – MASH, and by the Swiss National Science Foundation under grant 200021-124822 – VELASH.

References

1. Toscher, A., Jahrer, M., Bell, R.: The bigchaos solution to the netflix grand prize (2009), http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf
2. Gehler, P., Nowozin, S.: On feature combination for multiclass object classification. In: International Conference on Computer Vision (2009)
3. http://www.cs.toronto.edu/~kriz/cifar.html

Activity Recognition with Mobile Phones

Jordan Frank[1], Shie Mannor[2], and Doina Precup[1]

[1] School of Computer Science, McGill University, Montreal, Canada
{jordan.frank,dprecup}@cs.mcgill.ca
[2] Department of Electrical Engineering, Technion, Israel
shie@ee.technion.ac.il

Abstract. Our demonstration consists of a working activity and gait recognition system, implemented on a commercial smartphone. The activity recognition feature allows participants to train various activities, such as running, walking, or jumping, on the phone; the system can then identify when those activities are performed. The gait recognition feature learns particular characteristics of how participants walk, allowing the phone to identify the person carrying it.

1 Overview

We implemented a system for activity and gait recognition for the Google Android OS ©, which runs on smartphones from a variety of manufacturers. During our demonstration, we will allow participants to try out the various features, either demonstrating the phone's ability to recognize a pre-programmed set of activities (running, jogging, jumping, or walking), or the phone's ability to learn the characteristics of a participants' gait, or style of walking, and showing that the phone can later recognize that participant. Activity and gait recognition using wearable sensors are active research topics [see 8, 1, 9, for example]. Unlike previous work, we do not require specialized wearable sensors or careful placement of the phone on the participant's body. Our application works when the phone is simply placed in a trouser pocket. Additionally, the computation requirements for training these systems are prohibitive for real-time demonstrations, and in many cases even the classification requires more computation that can be afforded on a mobile device such as a smartphone. Our system was designed with computational efficiency in mind, and can both learn classifiers and perform classification in real-time on a low-powered mobile device.

The demonstration will be interactive, and participants will be asked to perform different activities, with the classification results displayed on a separate monitor for everyone to see.

2 Background

The activity and gait recognition systems are based on an algorithm called geometric template matching (GTM) [5] for comparing segments of time series. GTM operates by first building models, or templates, from short segments of labeled univariate time series, such as that collected by accelerometer sensors. These models are constructed

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 630–633, 2011.

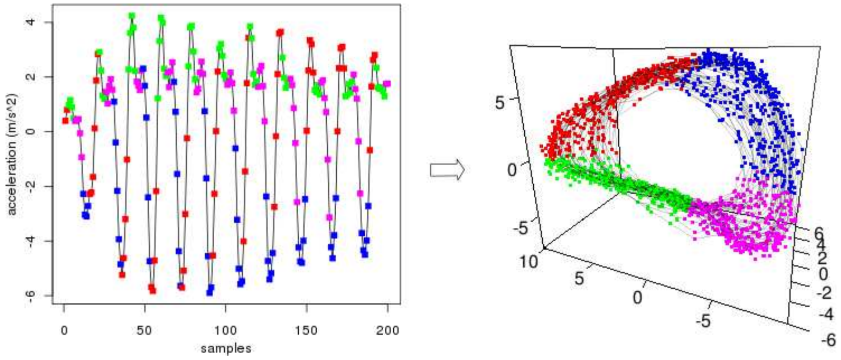


Fig. 1. An example of a time-delay embedding for a sequence of data collected from an accelerometer while a subject was riding a bicycle. The colours are purely for illustrative purposes and show that similar-looking segments of time series appear in similar regions of the embedded model.

using a technique called time-delay embedding, which is an approach developed in the nonlinear dynamical systems community for reconstructing dynamical models from measurements of some latent nonlinear system [6]. To compute the time-delay embedding for a sequence of measurements $o_1, \ldots, o_N$, one choses a lag parameter τ and a reconstruction dimension m, and then constructs the sequence of m-dimensional points:

$$x_i = (o_i, o_{i+\tau}, o_{i+2\tau}, \ldots, o_{i+(m-1)\tau}),$$

for $i = 1, \ldots, N - (m-1)\tau$. The model, then, consists of points, each composed of m τ-lagged samples of the data. The choice of m and τ are important, and while there are heuristics for choosing these values [3, 7], we have found that standard machine-learning approaches to parameter tuning, such as cross-validation and grid search are most effective. In addition, GTM is fairly robust with respect to the parameters, and values that work well for one problem (identifying running, for example), tend to work well for other problems (identifying biking or walking, for example). Figure 1 depicts the process of time-delay embedding for a segment of accelerometer data collected from an accelerometer sensor while a subject rode a stationary bicycle. It is clear from the embedding that, although the data is nonstationary, the periodic element is captured, and regions in the time-series that appear qualitatively similar are mapped to nearby points in the embedding.

The second step of GTM is to compare a new segment of time series to existing models. To do this, the new time series is projected into the embedding space and then a measure of similarity is computed by considering pairs of subsequent points as vectors and then computing a measure of similarity between the vectors representing the new time series and their nearest neighbours in the model. As a result, GTM computes a measure of similarity between the new time series and each of the models, and these similarity scores can be used to find the model that most closely matches the new time series, or as features for a more complex classifier. We have used these features to train an SVM [4] for classifying activities, and used the model with the highest similarity score for gait recognition [5]. In both cases, we achieve accuracies that exceed the performance of state-of-the art systems at a lower computational cost.

The advantages of GTM over other approaches for comparing segments of time series, such as dynamic time warping (DTW) [2] include its computational efficiency ($O(n \log n)$ to compute each similarity score, as opposed to $O(n^2)$ for DTW), and the fact that it can compare segments of different lengths.

3 Implementation

We have implemented a system for building models and performing classification for accelerometer data collected on mobile phones that run the Google Android OS ©. Due to the efficiency of the GTM algorithm, our system is able to build new models and perform classification in real-time on the phone without drastically reducing the battery life of the phone. The application that we will demonstrate provides an interface for collecting labeled training data from subjects, and for sending live classification results to a separate laptop for visualisation. The phone does not need to be affixed in a special way to the body of the subject, and can be placed in a trouser pocket. We intend to demonstrate two systems, one for recognizing particular activities, and one for recognizing individuals based on their gait[1].

For activity recognition, we will pretrain the phone with a number of activities that can be safely performed indoors without additional equipment, such as running, walking, jogging, and jumping up and down. Participants will then be asked to perform these activities, one at a time, while carrying the phone in their pocket. The classification will be performed on the phone, but to allow other participants to view the results, we will stream them wirelessly from the phone to a separate laptop that will display the results. In the event that participants come up with activities that we haven't thought to pretrain the phone with, the participants will be able to demonstrate the new activity while carrying the phone, and the phone will learn the new activity. Then, this new activity will be included in the set of activities that the phone can recognize.

For gait recognition, participants will be asked to walk with the phone for approximately 15 seconds, while the phone builds a model of their gait. The phone will be pretrained with the gait from 4 other subjects. The participant will be asked to walk again with the phone, and the phone will attempt to classify the particular person that is carrying it from the 5 current models. As with the activity recognition, the results will be streamed to a laptop, allowing others to view the results. As more participants perform the task, we will use their stored gaits, rather than the pretrained gaits.

4 Conclusion

We propose to demonstrate a state of the art activity and gait recognition system. The demonstration will be interactive, allowing participants to try out the systems, and the

[1] Given the nature of the data that the phone will be collecting, participants will be asked to sign a consent form. This demonstration will be performed in accordance with the McGill University Research Ethics Board, and our Certificate of Ethical Acceptability of Research Involving Humans will be clearly displayed. Any data that is collected will be anonymous, and will not be associated with any personally identifying information.

results will be displayed in real-time for other to see. These types of systems are typically evaluated in controlled environments, using special-purpose sensors, and often the processing is computationally expensive and done offline at a later time. This demo is interesting in that it uses consumer-grade mobile phones, does not require them to be carefully placed on a participant, and the processing is all done in real-time on the phone. There are many industrial applications for these types of systems, such as healthcare monitoring, fitness, and biometric security, making this demonstration appealing to both academic researchers and industry practitioners.

Additionally, the software is open-source and publicly available[2], and so participants that are interested in evaluating the system on their own can install the software on their own phones.

References

[1] Ailisto, H., Lindholm, M., Mantyjarvi, J., Vildjiounaite, E., Makela, S.: Identifying people from gait pattern with accelerometers. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 5779, pp. 7–14 (2005)

[2] Berndt, D., Clifford, J.: Using dynamic time warping to find patterns in time series. In: AAAI 1994 Workshop on Knowledge Discovery in Databases, pp. 229–248 (1994)

[3] Buzug, T., Pfister, G.: Optimal delay time and embedding dimension for delay-time coordinates by analysis of the global static and local dynamical behavior of strange attractors. Phys. Rev. A 45(10), 7073–7084 (1992)

[4] Cristianini, N., Shawe-Taylor, J.: An introduction to Support Vector Machines: and other kernel-based learning methods. Cambridge University Press, Cambridge (2000)

[5] Frank, J., Mannor, S., Precup, D.: Activity and gait recognition with time-delay embeddings. In: Proc. of the AAAI Conference on Artificial Intelligence (2010)

[6] Kantz, H., Schreiber, T.: Nonlinear time series analysis. Cambridge University Press, Cambridge (2004)

[7] Kennel, M., Brown, R., Abarbanel, H.: Determining embedding dimension for phase space reconstruction using the method of false nearest neighbors. Phys. Rev. A 45(6), 3403–3411 (1992)

[8] Lester, J., Choudhury, T., Borriello, G.: A practical approach to recognizing physical activities. In: Fishkin, K.P., Schiele, B., Nixon, P., Quigley, A. (eds.) PERVASIVE 2006. LNCS, vol. 3968, pp. 1–16. Springer, Heidelberg (2006)

[9] Subramanya, A., Raj, A., Bilmes, J., Fox, D.: Recognizing activities and spatial context using wearable sensors. In: Proc. of Uncertainty in Artificial Intelligence (2006)

[2] http://jwf.github.com/Humansense-Android-App/

MIME: A Framework for Interactive Visual Pattern Mining

Bart Goethals, Sandy Moens, and Jilles Vreeken

University of Antwerp, Belgium

Abstract. We present a framework for interactive visual pattern mining. Our system enables the user to browse through the data and patterns easily and intuitively, using a toolbox consisting of interestingness measures, mining algorithms and post-processing algorithms to assist in identifying interesting patterns. By mining interactively, we enable the user to combine their subjective interestingness measure and background knowledge with a wide variety of objective measures to easily and quickly mine the most important and interesting patterns. Basically, we enable the user to become an essential part of the mining algorithm. Our demo currently applies to mining interesting itemsets and association rules, and its extension to episodes and decision trees is ongoing research.

Keywords: MIME, Pattern Exploration, Interactive Visual Mining.

1 Introduction

Data mining is an inherently iterative process; the results of one analysis often lead to new questions, requiring more analysis. In an ideal world, this process is streamlined. That is, data mining is not only iterative, but also interactive: the user can give such feedback immediately, and easily browse the results. In traditional pattern mining, however, algorithms typically produce large amounts of patterns, many of which are not interesting to the user [8], and the results are typically only given in a flat text file, making it hard to analyze the results. By instead providing an iterative and interactive process, the user would be able to explore and refine the discovered patterns on the fly.

In this demo, we present a framework in which we allow the user to interactively mine a database for interesting itemsets and association rules. Our system visualizes all patterns discovered so far, yet, importantly lets the user interactively explore and dynamically modify these in an intuitive manner [2,9]. By the visualization, users can browse through the mined data more easily and so quickly discover important information. Essentially, in our framework visualization strengthens the interactive mining process, and vice versa.

Basically, our framework, MIME (Making Interactive Mining Easy), draws from user knowledge and interest to improve the collection of patterns discovered by the mining algorithms, by letting the user take control during the mining process, and allowing to adapt the results and so create useful collections of patterns. To assist the user, MIME offers an extensive toolbox of interestingness

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 634–637, 2011.

measures, mining techniques and visualizations. Using these, the user can easily identify and remove uninteresting or redundant patterns, extend or reduce existing patterns, or apply various post-processing techniques. As such, in our framework the user becomes an essential part of the mining algorithm.

2 Description of the System

We consider transactional (supermarket) databases where a transaction contains a number of items. Here, a pattern is an itemset or an association rule [7].

Most pattern mining techniques produce an amount of output that due to size is difficult to post-process. One could try to reduce the number of results by making the quality thresholds more strict. Unfortunately this does not guarantee the usefulness of the produced patterns. To this end, more is required.

MIME combines the knowledge of the expert and the computational strength of a computer to increase the probability of finding interesting patterns. This is achieved in a visual framework where the expert can create his own pattern collection. In order to evaluate the created/mined patterns, MIME implements many interestingness measures. This way the user can select a number of measures that are important for his purpose/domain. These measures improve the knowledge of the user. Our tool also contains a number of standard mining and post-processing algorithms such as Eclat and Apriori. The mining algorithms can be used to construct a starting set of patterns a user can play with and can further construct his collection of useful patterns from.

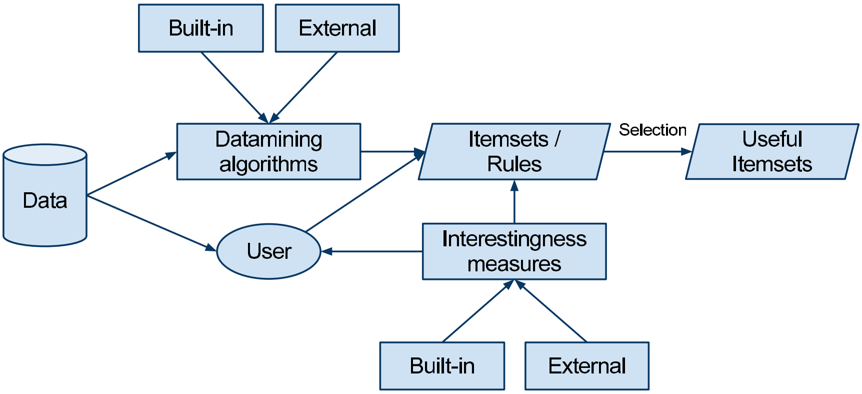


Fig. 1. MIME Workflow

The overall workflow of MIME is shown in Figure 1. Data is read from a database file and can be used by a user or by mining algorithms to create rules and itemsets. From the created patterns a user can make a decision which patterns are useful, based on the provided interestingness measures. The user may also apply post-processing algorithms to the created collection of patterns.

An important aspect of MIME is the way in which the user interacts with the current state of the system. All interactions made by the user have a direct impact on its current state (i.e. the items, itemsets and association rules). Thereto MIME contains 4 different panes a user can interact with: the *source dock*, the *work dock* (or *workspaces*), a *toolbox* and the *global overview*. The source dock visualizes items in the dataset. All items are ranked in decreasing order based on an active interestingness measure. The work dock shows mined patterns. Patterns can be altered, deleted, expanded, etc. The toolbox contains measures, mining algorithms and post-processing algorithms which can be applied. The global overview provides more general information about the workspaces.

All information provided by MIME is computed on-the-fly. When selecting new measures from the toolbox, it computes the corresponding ranks of the discovered patterns and shows them as soon as they are available. Extending or reducing patterns also reflects in an immediate recalculation of the applied measures. The nature of any-time algorithms applies to our framework in the sense that partial information is shown when available. Therefore, we have made extensive use of caching and threading in our system. Using caching, MIME minimizes the number of computations that are made by reusing previously computed information. Threading is used to compute information in the background.

An important feature is the *Best Pattern Extension*. This feature allows the user to immediately see the individual impact of remaining items on an existing pattern, i.e. a pattern is repeatedly extended by one of the remaining items in the dataset and the ranks (based on the active measure) are computed and shown in the source dock, again using rank decreasing order. Other functionalities for generating patterns based on existing patterns, are the generation of all subsets, subsets of given size, closed supersets, etc. Mining algorithms can also use discovered patterns to filter the mining output. All these features (including mining and post-processing algorithms) provide the user with a toolbox for easy and fast generation of possibly useful patterns.

From the workflow notice the *External* nodes indicating the use of external/plugin functionality in the tool. In order to provide a widely applicable and easily extensible tool, we have equipped our tool with a plugin system, such that existing mining and post-processing algorithms can be used. The produced patterns are automatically loaded into the MIME framework.

A demonstration video introducing the basic functionalities of MIME can be found on the webpage of our research group[1].

3 Related Work

A lot of work has been done comparing and evaluating different objective interestingness measures [6,10]. In our tool several objective interestingness measures have been incorporated, but it is the combination of user-knowledge and objective measures enabling subjective interestingness criteria to be applied.

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

Also in the context of Inductive Databases several interactive constraint-based mining frameworks have been studied [3]. Our framework could be built on top of such an inductive database implementation.

Most similar to the system presented here, Ankerst et al. proposed a framework for mining decision trees [1]. The user and computer work together in this process such that in each step either the user can make the decision for a new split or the computer can make this decision. The computer also provides extra computational power by showing the best split, look-ahead information, purity measures etc. Our approach is similar, but applies to frequent itemsets and association rules, instead of decision trees.

Some methods for visualizing patterns exist [4,5]. These visualizers present the output of mining algorithms in a compact and graphical format, and allow to further filter the output using queries. They do not provide means to mine the database interactively using subjective criteria and also do not allow to further explore existing patterns.

References

1. Ankerst, M., Ester, M., Kriegel, H.-P.: Towards an effective cooperation of the user and the computer for classification. In: Proc. ACM SIGKDD, pp. 179–188 (2000)
2. Bertini, E., Lalanne, D.: Investigating and reflecting on the integration of automatic data analysis and visualization in knowledge discovery. SIGKDD Explor. Newsl. 11, 9–18 (2010)
3. Blockeel, H., Calders, T., Fromont, E., Goethals, B., Prado, A., Robardet, C.: An inductive database prototype based on virtual mining views. In: Proc. ACM SIGKDD, pp. 1061–1064 (2008)
4. Leung, C.K.-S., Carmichael, C.L.: FpVAT: a visual analytic tool for supporting frequent pattern mining. SIGKDD Explor. Newsl. 11, 39–48 (2010)
5. Techapichetvanich, K., Datta, A.: VisAR: a new technique for visualizing mined association rules. In: Li, X., Wang, S., Dong, Z. (eds.) Adv. Data Min. Appl., pp. 728–728. Springer, Berlin (2005)
6. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A survey. ACM Comput. Surv. 38 (September 2006)
7. Goethals, B.: Frequent set mining. In: Data Mining and Knowledge Discovery Handbook, pp. 321–338 (2010)
8. Vreeken, J., Tatti, N., Goethals, B.: Useful patterns (UP'10) ACM SIGKDD workshop report. SIGKDD Explor. Newsl. 12, 56–58 (2011)
9. Keim, D.A.: Information visualization and visual data mining. IEEE Trans. Vis. Comp. Graph. 8, 1–8 (2002)
10. Tan, P.-N., Kumar, V., Srivastava, J.: Selecting the right interestingness measure for association patterns. In: Proc. ACM SIGKDD, pp. 32–41 (2002)

InFeRno – An Intelligent Framework for Recognizing Pornographic Web Pages

Sotiris Karavarsamis, Nikos Ntarmos, and Konstantinos Blekas

Department of Computer Science, University of Ioannina,
PO Box1186, 45110 Ioannina, Greece
{cs061205,ntarmos,kblekas}@cs.uoi.gr

Abstract. In this work we present InFeRno, an intelligent web pornography elimination system, classifying web pages based solely on their visual content. The main characteristics of our system include: (i) a powerful vector space with a small but sufficient number of features that manage to improve the discriminative ability of the SVM classifier; (ii) an extra class (*bikini*) that strengthens the performance of the classifier; (iii) an overall classification scheme that achieves high accuracy at considerably lower runtime costs compared to current state-of-the-art systems; and (iv) a full-fledged implementation of the proposed system capable of being integrated with ICAP-aware web proxy cache servers.

1 Introduction

Despite the usefulness and ease-of-access to a plethora of information scattered on the web, the Internet has become a hostile environment for unprotected people like children. Pornography is considered as sensitive information that is believed to be harmful for some groups of people. In the course of autonomously discriminating and blocking access to such content, Forsyth et al [1] was the first to implement a system comprising a figure grouper that inferred the existence of nude human figures. However, a disadvantage of this method is its high processing time; typically it takes about 6 minutes to process a suspect image [2], that makes it impractical in real-world applications. Wang et al. [3] proposed another pornography elimination system called WIPE^{TM}, which employed Daubechies wavelet analysis and extraction of invariant central moments.

Pornography filtering has also recently been applied to web searching upon user queries request. For instance, Rowley et al [4] have proposed a method for identifying nude images which has been part of Google^{TM} safe search. Also, Hu et al. [5] proposed a system to recognize pornographic web pages by using both text and images information. It employs a combination of discrete and continuous text classifiers, a nearest-neighbor image classifier, as well as a method for fusing both kind of information. Similarly, the POESIA project [2] is an open-source system for blocking pornography, which applies a combination of image and harmful-symbol filtering and also text classification via NLP techniques.

In this work we present InFeRno, an intelligent web pornography elimination system, classifying web pages based solely on their visual content. The main

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 638–641, 2011.

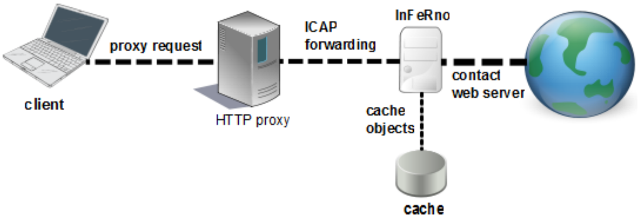


Fig. 1. InFeRno architecture

characteristics of our system include: (i) a powerful vector space with a small but sufficient number of features that manage to improve the discriminative ability of the SVM classifier; (ii) an extra class (*bikini*) that strengthens the performance of the classifier; (iii) an overall classification scheme that achieves high accuracy at considerably lower runtime costs compared to current state-of-the-art systems; and (iv) a full-fledged implementation of the proposed system capable of being integrated with any ICAP[1]-aware web proxy cache server.

2 System Architecture

The InFeRno core is implemented as an ICAP module, capable of communicating with any ICAP-enabled HTTP proxy software (see Fig 1). Client requests for web objects (pages or images) are forwarded to the proxy infrastructure which then delegates them to the InFeRno module. The latter first checks its local cache for an existing classification result. If one is found, it is returned to the client via the HTTP proxy; in the opposite case, InFeRno needs to fetch the requested object from the remote web server and classify it. For individual images, the result is returned to the client once the classification is over. In the case of web (html) pages, InFeRno further prefetches and classifies in parallel all (or a sample of) images referenced in the page code, then fuses the individual results to produce an overall classification for the web page. As image fetching and classification is done in parallel, the time required to classify a web page is very close to the time required to classify the "hardest" of its images (the overhead of the subsequent fusion step is in the order of a few milliseconds). Along with the caching of fetched objects and classification results, our system adds a slight overhead (of on average no more than 2") in user-perceived web page loading times.

InFeRno can classify and filter content at various levels, at the discretion of its administrator: (i) at the web page level, it produces assessments for complete web pages (including all referenced images) and can deny access if the page is deemed pornographic (returning an appropriate error page to the client), and (ii) at the image level, it can allow web pages to be rendered at the client, but deny access to individual images if they are deemed pornographic (returning a blurred out version of the image). The administrator can further tweak the

[1] ICAP stands for *Internet Content Adaptation Protocol* and is described in RFC 3507

acceptance thresholds for the amount of pornographic content being permitted per web page, thus allowing for an even more flexible configuration. The system core can also be integrated with other services, such as email scanner daemons, network firewalls, etc., providing image classification and porn filtering to a diverse array of end-user applications. Last, InFeRno can be configured either to not log client info so as to achieve a certain level of anonymization and privacy, or for full logging as an extra deterrent against illicit content access.

3 Classification System

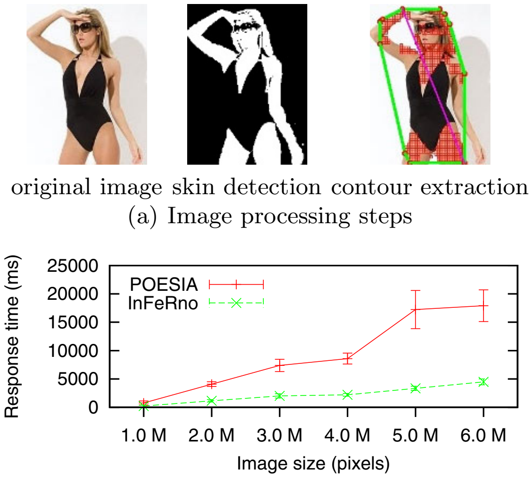
Skin detection. InFeRno employs a simple rule-based skin color detection technique which uses deterministic rules imposing relations between the R, G and B color channels [6]. As experiments have shown, this technique provides a satisfactory performance. However, it provides many false positives in the presence of either excessive illumination or of objects with skin-like color. For this reason, we have used an adaptive Gamma correction method, that eliminates the effect of this phenomenon and further improves the overall performance.

Contour extraction. In order to extract the contour of human nudes, we have followed a geometric approach, presented in [7], that uses a region splitting scheme. Initially, the image plane is partitioned into four equal quadrants. Subsequently, the corresponding skin and non-skin pixel intensity histograms in each quadrant are first found, and then two measurements are calculated: the *skin to non-skin ratio* and the *kurtosis*. Next, we decide to further split a region if both of the above measures exceed two predefined thresholds, estimated via extensive experimentation with diverse datasets. The outer corners of the last constructed connected split-regions are regarded as control points which are finally used to establish the convex hull and localize the Region of Interest (ROI) in the image.

Feature extraction and Classifier. The next step is to extract a set of features regarding the size, orientation, and content of the contour that has been estimated previously. We have selected to calculate 15 such features, which are:

- Mean values and variances of the RGB color channels of all non-skin pixels.
- Ratio of the total skin and non-skin pixels delimited by the contour.
- 7 invariant spatial Hu moments of the surface.
- The angle of the diameter of the convex hull.

The above features are combined to form a feature vector space, which is used next for classifying images. We have selected the Support Vector Machine (SVM) as a classifier, using an one-against all scheme and simple linear kernels, thus avoiding any free parameters. An innovation of our classification scheme is that we have considered one more class apart from pornographic and non-pornographic: the *bikini* class of people wearing bikinis, swimsuits, and so forth. Experiments have shown that introducing this third class manages to significantly not only increase the total performance of the classifier, but also strengthen its decision. Furthermore, the bikini class can optionally act as


original image skin detection contour extraction

(a) Image processing steps

(b) InFeRno vs. POESIA: response time in terms of image size

Fig. 2. Operation and performance of InFeRno

an extra level of nudity (a *softer* version of "porn") recognized by the system. Fig 2(a) shows an example of the basic processing steps of our system.

For training the multi-class SVM, we have created a dataset of 5680 images (660 pornographic, 700 bikini and 4320 assorted benign images), that have been manually labeled. Using 10-fold cross validation we tested our classifier and received an accuracy of 98 %, 97 % and 98.8 % for our three classes, respectively. Last, Fig 2(b) depicts the time required to classify single images by both our system and the POESIA image classifier. Due to our system architecture and selected feature set, InFeRno achieves an equally high accuracy classification, but at a portion (more than a 4× speedup) of the time required by POESIA.

References

1. Fleck, M., Forsyth, D., Bregler, C.: Finding naked people. Computer Vision, 593–602 (1996)
2. POESIA project, http://www.poesia-filter.org/
3. Wang, J.Z., Li, J., Wiederhold, G., Firschein, O.: System for screening objectionable images. Computer Communications 21, 1355–1360 (1998)
4. Rowley, H., Jing, Y., Baluja, S.: Large scale image-based adult-content filtering. In: Int. Conf. on Computer Vision Theory and Applications (2006)
5. Hu, W., Wu, O., Chen, Z., Fu, Z., Maybank, S.: Recognition of pornographic web pages by classifying texts and images. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1019–1034 (2007)
6. Vezhnevets, V., Sazonov, V., Andreeva, A.: A survey on pixel-based skin color detection techniques. In: Graphicon, pp. 85–92 (2003)
7. Yang, J., Shi, Y., Xiao, M.: Geometric feature-based skin image classification. In: Int. Conf. on Adv. Intell. Comp. Theories and Applications, pp. 1158–1169 (2007)

MetaData Retrieval: A Software Prototype for the Annotation of Maps with Social Metadata

Rosa Meo[1], Elena Roglia[1,2], and Enrico Ponassi[1]

[1] Dep. Computer Science, University of Torino, Italy
[2] ITHACA, Torino

Abstract. MetaData Retrieval (MDR) is a software module for the enrichment of geo-referenced maps with metadata. Metadata are annotations on spatial locations that are taken from the Volunteered Graphical Information projects like OpenStreetMap and GeoNames.

The MDR user acts with a user-friendly GUI, a Query By Example in which the user specifies in a multi-dimensional data model the spatial objects for which new information are searched for. The request is translated into SQL queries for the database and in web service requests for OpenStreetMap and GeoNames. Downloaded annotations are checked and compared with the history for duplicate elimination. Annotations are presented to the user in the context of an interactive, geo-referenced map and in a hierarchical, ontological structure, that is a facility for indexing and browsing. On demand, an annotation is stored in the system history. Finally, the user can filter the annotations that characterize a specified area by a statistical filter that compares the annotation frequency with the neighborhood.

1 Introduction to MDR and Comparison with Related Works

The natural disasters require territory monitoring. SMAT is a distributed system for territory monitoring by means of Unmanned Aircraft Vehicles (UAVs). A UAV is equipped with payload sensors that download video of the target territory in the system.

The system should support the work of multiple operators. The operators can provide additional maps with annotations, metadata and accompanying files extracted from external sources (such as the web).

The software prototype that we describe in this paper is MetaData Retrieval (MDR). It is a geo-spatial web service integrated in the SOA architecture of the SMAT system. The main focus of MDR is to provide additional information on the locations included in cartographic maps, referred to as metadata. The availability of up-to-date cartographic maps is one of the main motivations of MDR because the maps get soon outdated. In addition, cartographic maps are often thematic and do not contain all the information that is needed by any user. On the contrary, there exists on the web a large amount of information on the geographical areas generated by Volunteered Graphic Information (VGI) projects by the everyday experience of people and by the integration of different cartographic databases: OpenStreetMap [3] (a free editable map of the whole world) and GeoNames [2] (the description and definition of over eight million named locations).

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 642–645, 2011.

From this viewpoint, MDR is an innovative work because it integrates the information from VGI in a territory surveillance project by means of UAVs. For territory monitoring there is the need to retrieve and present the history of the annotations on a certain area of interest, to interactively explore the annotations by category, time, spatial object and compare them with the current status of the area. This exploratory interaction makes emerge easily the differences between the current geographical area and the past. This procedure is not present in VGI projects like LinkedGeoData [1] which work on the ontology for the annotations but do not provide a temporal view of the annotations in a selected area.

The information on locations is checked by MDR to be in a well-formed format and not redundant w.r.t. the history (i.e., containing no duplicates). In addition, annotations can be filtered by the user with a statistical filter that is based on the assumption of spatial auto-correlation. Significant annotations are those ones whose frequency in the area is an anomaly w.r.t. the frequency distribution in the neighborhood. In this way, it gives a characterization of the selected area and guarantees an increased level of reliability against noise and users errors. In this respect MDR is again a novelty w.r.t. the state of the art. For instance, [4] searches for frequent patterns in annotations but it does not filter annotations by outlier detection. [5] characterizes an area with a selection of the spatial features but it is much more difficult to tune because it considers three nested areas and user defined thresholds.

2 MDR Description

MDR is based on the annotations present in OpenStreetMap (OSM) and GeoNames. OSM maps are made up of three basic elements that form an ontological description: nodes, ways and relations. Each element has an arbitrary number of properties (tags) which are key-value pairs: ’key’ represents a broad concept; ’value’ is a specialization (like in key=historic; value=monument). A node represents a map feature or a standalone entity: it consists of the latitude and longitude of the location in the geocentric coordinate system, the user name who provided the data, a time-stamp and of additional feature attributes specified by key-value pairs.

GeoNames provides location descriptions which consist in the official name, a general description, demographic data, images, etc. Similarly to OSM, it provides a set of web services that provide an XML file with the annotations of the selected area.

Figure 1 is a screen-shot of MDR with its main output: an annotated, interactive cartographic map. The left-hand side of the window displays a tree-like arrangement that helps the user to index and browse the annotations. They are ordered first by location, then by retrieval time and finally by category (key). Each annotation on the left corresponds to an icon that can be correctly geo-referenced in the map on the right. The tree is navigable: the user can choose a specific spatial object, a point in time and a specific annotation category in the tree. As a consequence the relative annotations are correctly positioned on the map. If the user chooses a specific annotation from the list, a message box opens showing descriptive information on the location.

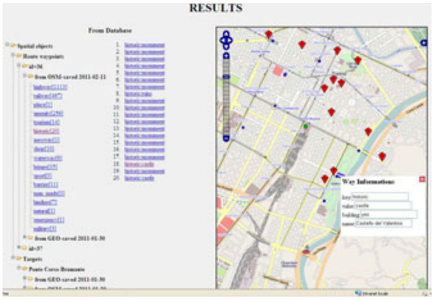


Fig. 1. Interactive map with annotations from the historical database

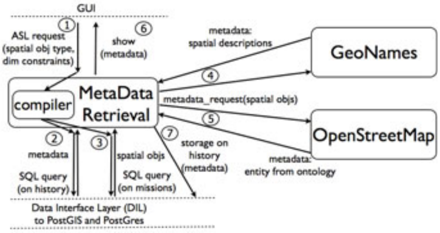

GUI
1
ASL request (spatial obj type, dim constraints)
6
show (metadata)
compiler
MetaData Retrieval
metadata: spatial descriptions
4
GeoNames
metadata_request(spatial objs)
5
OpenStreetMap
metadata: entity from ontology
2
metadata
SQL query (on history)
3
spatial objs
SQL query (on missions)
7
storage on history (metadata)
Data Interface Layer (DIL) to PostGIS and PostGres

Fig. 2. Software architecture for MDR

2.1 Architecture Description

In order to accomplish the SMAT project goals, SOA is the suitable architectural choice because it allows the integration of different independent systems and services. Probably the best known spatial web services are Google Maps/Earth. In MDR, however, we chose to use open source software components (like Geoserver, OpenLayers for map visualization and development and PostGIS for the spatial database) that adopt open standards. Figure 2 shows the software architecture.

Any user's request to MDR searches for the metadata of some specified spatial objects. The spatial objects are involved in some of the facts contained in the system, general-purpose, multi-dimensional data model. In the case of the demo, the data model represents UAVs missions that are stored in the SMAT system database. The missions facts are described by the following, independent dimensions: UAV, sensor, target, airport, mission execution time, metadata time, space (spatial coordinates).

In MDR a user-friendly GUI allows the user to specify for which spatial objects involved in missions the annotations are requested. It is a sort of Query By Example which shows each dimension and allows the selection of any combination of dimensions values that allows the identification of the spatial objects of interest. The value selection is performed in a smart way because a drop-down list is presented with meaningful values. As regards the output annotations, the user can specify by a drop-down list the maximum distance (in Km) allowed between the spatial objects and the nearby locations whose annotations will be displayed. The dimension constraints generate a specification in an Abstract Specification Language (ASL) whose meaning is: *Retrieve the metadata associated to the specified spatial objects involved in the missions satisfying all the constraints*.

The MDR compiler translates the ASL query into a set of SQL queries for the database: the first type of SQL query retrieves the spatial objects satisfying the mission constraints and returns the spatial coordinates of the locations necessary to perform a web service request to OSM and GeoNames. The second query returns the metadata already stored in the system database for the same spatial objects. The complexity of the generation of a SQL query is linear in the number of constrained dimensions, being them independent as dimensions of a data warehouse star schema.

The answer of OSM and GeoNames is constituted by an XML file for each spatial object. Each file is parsed and the identified tags are presented in the output page grouped by category and ordered by time. Figure 1 presents the results for a query asking the annotations around the bridges over the river Po in Torino. Annotations show the historic monuments as well as other categories (railway, highway) and the categories corresponding to public services (like parkings, shops, amenities).

3 Geographic Characterization

We describe here how we obtain the content characterization of a geographical area. The characterization occurs in terms of the concepts corresponding to the tag categories provided as annotations. The filter consists in the extraction of the tags that are significant by a statistical validation method. It compares the frequency of each tag encountered in the given area, with the frequency distribution of the tags of the same category in the surrounding geographical areas.

We sketch the way in which we draw the sample from which we generated the frequency distribution. We build a regular grid composed of a total of 49 cells, surrounding the central area. All the cells of the grid have equal surface area of the central, target cell: thus in any cell each tag category has the same probability to appear. The frequency of each tag category appearing in the central cell is collected also in all the neighborhood areas: their distribution in the neighborhood is generated and compared with the frequency in the central cell. We perform a standard, statistical test on the frequency of each category with a very low significance level ($\alpha = 0.001$). The frequencies of the tag categories in the central area that are outliers of the frequency distribution in the surrounding areas are highlighted as the interesting ones because surprising. In fact, given the property of spatial auto-correlation of the features, most of the tag categories are expected to occur also in the neighborhood with a frequency similar to the central cell. The tags that are selected are: (i) the tags on which the majority of the users agree; (ii) the tags that characterize the area because discriminate it with the surroundings. As an example for the central area of Torino, significant tags result the fountains, the restaurants, the tram stops, the bicycle rental sites which confirm the characterization of the touristic area in contrast with the residential neighbourhood.

References

1. Auer, S., Dietzold, S., Jens, L., Hellmann, S., Aumueller, D.: Triplify lightweight linked data publication from relational databases. In: Proc. of WWW (2009)
2. GeoNames, http://www.geonames.org (Retrieved 10-11-2010)
3. Haklay, M., Weber, P.: Openstreetmap: User-generated street map. IEEE Pervasive Computing 7(4), 12–18 (2008)
4. Sengstock, C., Gertz, M.: Exploring volunteered geographic information using scale-dependent frequent pattern mining. In: Proceedings of GIScience (2010)
5. Tomko, M., Pulves, R.: Venice, city of canals: Characterizing regions through content classification. Transactions in GIS 7, 295–314 (2009)

TRUMIT: A Tool to Support Large-Scale Mining of Text Association Rules

Robert Neumayer[1], George Tsatsaronis[2], and Kjetil Nørvåg[1]

[1] Norwegian University of Science and Technology,
Department of Computer and Information Science, Trondheim, Norway
{neumayer,noervaag}@idi.ntnu.no
[2] Biotechnology Center
Technical University of Dresden, Germany
george.tsatsaronis@biotec.tu-dresden.de

Abstract. Due to the nature of textual data the application of association rule mining in text corpora has attracted the focus of the research scientific community for years. In this paper we demonstrate a system that can efficiently mine association rules from text. The system annotates terms using several annotators, and extracts text association rules between terms or categories of terms. An additional contribution of this work is the inclusion of novel unsupervised evaluation measures for weighting and ranking the importance of the text rules. We demonstrate the functionalities of our system with two text collections, a set of *Wikileaks* documents, and one from TREC-7.

1 Introduction

Association Rule Mining (*ARM*) is a well-researched field of data mining. Rules can help to uncover hidden or previously unknown associations. A rule in the form of $A => B$, denotes an implication of element or item B by item A. Association rules have successfully been used in a wide range of domains, e.g., market basket analysis, law enforcement, biotechnology.

Lately, the benefits of applying ARM to text have appeared in query refinement and applications in text search and has become an objective of vital interest to the area of text mining and the practitioners of the field. *Text Association Rule Mining* (*TARM*) faces new challenges with respect to the volume of the data and the number of distinct items. Another major challenge is the interpretation of the rules as well as the evaluation and ranking of these rules according to their importance.

In this paper we address those issues, and we present the *text rule mining testbench* (TRUMIT). Our system implements all the stages of *TARM*, namely pre-processing, the actual mining of rules, and thorough analysis and visualization of the generated rules. We give a special focus on the pre-processing, and more precisely on the annotation step. We integrate a range of different annotation types including simple tokenization and matching, part-of-speech (POS) tagging, named entity recognition (NER), or more advanced types of semantic annotation based on the WordNet[1] and the OpenCalais[2] cat-

[1] http://wordnet.princeton.edu/

[2] http://www.opencalais.com/

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 646–649, 2011.

egories. The input text may be filtered according to the different annotation types. After pre-processing, the system allows for *Frequent Pattern Mining* (*FPM*) via the *Hadoop Map Reduce framework* before we extract the actual rules. Finally, we integrate existent, but also new, evaluation metrics for the extracted text rules. The system provides an easy-to-use interface for inspecting the generated rules, which may also function as a search interface for the respective collection.

2 Text Association Rule Mining

In its original form, association rule mining discovers regularities in data [1]. We consider a set of transactions $D = \{d_1, d_2, ..., d_n\}$, each transaction $d_i \in D$ comprises a set of items, i.e., $d_i = \{i_1, i_2, ..., i_m\}$. We also denote with I the set of all distinct items i_j that may occur in any transaction $d_i \in D$. Any subset $I_m \subseteq I$ is called an *itemset*.

In our context we denote a text document as T_i which belongs to a document collection T; we define as D the set of all text sentences. The set of all possible items of a transaction is the set of distinct terms T. Additionally, we create a second level of items, comprising all of the annotations A of all terms $t_j \in T_i$. Examples of such annotations may be *Person*, *Date*, or *Company*, which generates category rules of the form *Person* => *Company*, or *Company* => *Date*. Extraction of such rules unfolds the meaning of *Bill Gates* => *Microsoft*, or *Google* => *1998*. We include annotations provided by *OpenCalais* and we consult the *WordNet* thesaurus to annotate nouns with their respective domain terms. However, the system's architecture allows for easy integration of new annotation plug-ins. An example category rule, by including *WordNet* domain terms, could be *Animal* => *Company*, which might denote the information that *Company* conducts animal experiments in the life sciences domain.

With regards to related work, an approach for activity and emotion rule mining is presented in [3]. The authors mine simplified rules from a large collection of blog entries based on multiple minimum support. A tool to mine *maximal association rules* is introduced in [2]. A limited set of named entities is used for association rule mining. Experiments are performed on collections up to 10.000 documents. Another toolkit for *TARM* is presented in [4]. The authors worked on ARM in temporal document collections, and extended previous work by performing mining based on semantics, as well as by studying appropriate evaluation techniques. The focus was on the temporal aspect of the extracted rules. Scalability issues were not taken into account to a satisfactory extent.

To the best of our knowledge there does not exist a tool that includes all of the main *TARM* stages, shown in Fig. 1, which are included in our system. Existing tools are either not focused on text, or lack a capable user interface, or they cannot offer generic annotation integration, or are not competitive in terms of scalability. We aim to satisfy all of these requirements.

3 TRUMIT: Text Rule Mining Testbench

TRUMIT comprises four distinct processing stages, shown in Fig. 1. All intermediate results are stored on disk making it easier to work with large scale collections. In the following we will describe the processing stages of TRUMIT.

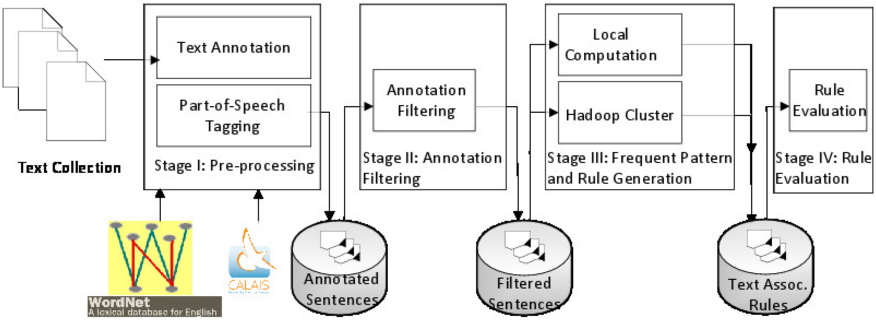


Fig. 1. Overview of the system including its four processing stages

Text annotation: This stage integrates a wide range of annotation plugins based on *Apache UIMA*. Currently, the following annotator types are supported: *language annotator*, *open calais annotator*, *POS annotator*, *Stanford NER annotator*, *Wordnet domain annotator*[3], *maui keyword annotator*[4], *lexical emotion annotator*, and *Wikipedia miner annotator*[5]. For reasons of simplicity in the figure we only demonstrate two of the used annotators (*WordNet* and *OpenCalais*). The system architecture at this stage is general enough to host any other annotator for the pre-processing step, through *Apache UIMA*.

Annotation filtering: At this second stage, we allow for flexible filtering of annotations by the user. Filtering is possible according to certain POS tags, entities, keywords or any other annotation technique used in the pre-processing step.

Frequent pattern mining and rule generation: The output of the filtering stage can subsequently be used for frequent itemset generation. To this end, a *Hadoop* job for the *fp-growth* implementation of the *Apache Mahout*[6] library is started and, either computed locally or sent to a map reduce cluster. In this way we can guarantee the computation without being restricted to the hardware configuration of the client machine. From the result of the *map reduce* job we generate rules and assign scores based on *rule interestingness* or *evaluation criteria* that our system supports. Currently, the system includes *confidence*, *support*, *semantic relatedness*, and *similarity variance*.

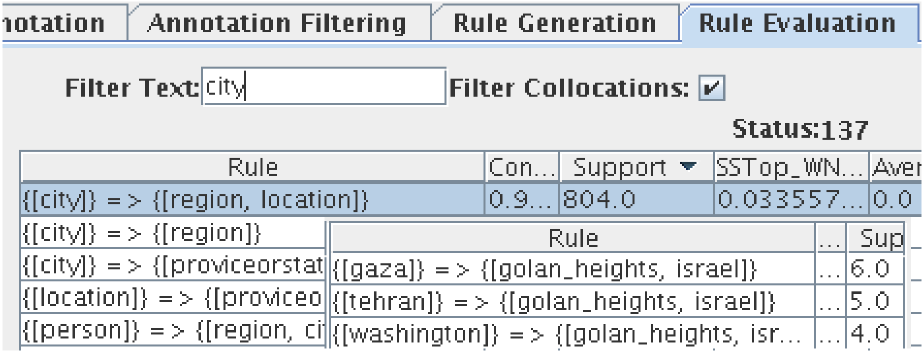
Rule evaluation: Once the text association rules are computed and scored, in this final stage we provide an interactive way to the user of browsing and analysing them. Through component, the rules can be sorted according to the evaluation criteria described previously. We also provide means to easily search for documents matching certain rules. We show an example of how category rules can be mapped to rules based on text only in Fig. 2.

[3] http://nlp.stanford.edu/software/CRF-NER.shtml

[4] http://code.google.com/p/maui-indexer

[5] http://wikipedia-miner.sourceforge.net/

[6] http://mahout.apache.org



Filter Text: city

Filter Collocations: ☑

Status:137

Rule	Con...	Support ▼	SSTop_WN...	Aver
{[city]} => {[region, location]}	0.9...	804.0	0.033557...	0.0
{[city]} => {[region]}				
{[city]} => {[proviceorstat				
{[location]} => {[proviceo				
{[person]} => {[region, ci				

Rule	...	Sup
{[gaza]} => {[golan_heights, israel]}	...	6.0
{[tehran]} => {[golan_heights, israel]}	...	5.0
{[washington]} => {[golan_heights, isr...	...	4.0

Fig. 2. TRUMIT user interface

4 Demonstration

In this demo we will show the full workflow by the example of the *cablegate* collection which is currently released by *wikileaks*. To this end we will illustrate how annotation can be performed with a range of plugins. Further we show the integration of both local rule generation and the map reduce framework for frequent pattern mining. We will show the benefits of additional evaluation measures for text association rules by example. Additionally, we will show analysis of rules which were computed offline for the 500.000 document TREC7 ad-hoc collection. The demonstration shows that our testbench offers a helpful tool integrating state-of-the-art libraries and technologies in its back-end.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. SIGMOD Record 22, 207–216 (1993)
2. Amir, A., Aumann, Y., Feldman, R., Fresko, M.: Maximal association rules: A tool for mining associations in text. Journal of Int. Inf. Systems 25(3), 333–345 (2005)
3. Kurashima, T., Fujimura, K., Okuda, H.: Discovering association rules on experiences from large-scale blog entries. In: Boughanem, M., Berrut, C., Mothe, J., Soule-Dupuy, C. (eds.) ECIR 2009. LNCS, vol. 5478, pp. 546–553. Springer, Heidelberg (2009)
4. Nørvåg, K., Fivelstad, O.K.: Semantic-based temporal text-rule mining. In: Gelbukh, A. (ed.) CICLing 2009. LNCS, vol. 5449, pp. 442–455. Springer, Heidelberg (2009)

Traffic Jams Detection Using Flock Mining

Rebecca Ong, Fabio Pinelli, Roberto Trasarti, Mirco Nanni,
Chiara Renso, Salvatore Rinzivillo, and Fosca Giannotti

Pisa KDD Laboratory,
ISTI - CNR, Italy

1 Introduction

The widespread use of GPS devices on cars enables the collection of time-dependent positions of vehicles and, hence, of their movements on the road network. It is possible to analyze such huge collection of data to look for critical situation on the traffic flow. The offline analysis of traffic congestions represents a challenging task for urban mobility managers. This kind of analysis can be used by the traffic planner to predict future areas of traffic congestions, or to improve the accessibility to specific attraction points in a city. Many traffic systems adopt *ad-hoc* sensors like cameras, induction loops, magnetic sensors to monitor the status of the traffic flows: these systems are very expensive for installation and maintenance, and they are restricted to the local monitoring of the road arcs where they are installed. On the contrary, the use of GPS data to check the traffic conditions requires low installation costs (a part for the installation on the vehicle) and it enables to virtually monitoring the entire road network.

In this demo we present an innovative tool that exploits the data collected from GPS-enabled cars to detect the occurrences of traffic jams on the road network. The detection of potential traffic jams is based on the discovery of slowly moving *flock patterns*, i.e. a set of objects slowly moving together for a minimum amount of time [2,1,5]. The tool has been integrated in the M-Atlas system [4,7] exploiting the implementation of the T-Flock algorithm provided by the system. To the best of our knowledge this is the first system that uses GPS data, combined with flock mining, to detect traffic congestions. Most of the approaches available in the literature for traffic analysis are based on aggregation of spatial or temporal data focusing on predefined areas [3]. It is important to point out that this tool does not provide real time analysis, but instead it allows the analysis of the historical data. We will sketch here a case study on a dataset of around 40,000 GPS-tracked cars in the surrounding of Pisa in Italy. The demo will highlight the tight integration of the spatio-temporal and data mining tools of the M-Atlas system and the graphical user interface that assists the DM analyst in driving his/her analysis.

2 Problem Definition

We define a *traffic jam* as a group of vehicles moving close and slowly for a minimum period. Following the definition given in [6], a *T-Flock* is formally described as:

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 650–653, 2011.

Definition 1 (T-Flock). *Given a set of n trajectories where each trajectory consists of τ line segments a flock in a time interval $[t_i, t_j]$, where $j - i + 1 \geqslant k$ consists of at least m entities such that for every discrete time step t_l, $i \leqslant l \leqslant j$, there is a disk of radius r that contains all the m entities.*

According to Definition 1, a T-Flock represents a generic movement of different entities moving together. For the objectives of our analysis, we enrich the T-Flock definition with a speed threshold, ensuring that the discovered patterns do not move more quickly w.r.t. the given threshold.

Definition 2 (T-Flock Jam). *Given a speed threshold s, a T-Flock T is a T-Flock Jam if its speed is lower than s.*

We propose a new method for traffic jam detection based on the combination of the *(1)* data mining query language and T-Flock algorithm implementation provided by M-Atlas, and *(2)* a graphical user interface to refine and interpret the results. Initially, the M-Atlas system is used to extract the T-Flock patterns from a dataset. The extracted patterns are rendered on the screen by means of visual metaphors to highlight relevant properties of the flocks, like support, velocity, duration. The visual interface allows also the definition and the combination of constraints on the patterns, in order to select the subset of T-Flocks patterns that satisfy the T-Flock Jam definition. The integration of the methodology within the M-Atlas system allows the transparent integration of the analytical process with the other tools provided by the system.

3 The Case Study

We briefly sketch here an analytical case study performed on a set of GPS-tracked vehicles in the Pisa surroundings.

The Dataset. The Pisa dataset contains a set of timestamped points of the form (*id*, *lat*, *long*, *t*) from around 40,000 cars, which represents 2% of registered cars in the coastal areas of Tuscany. These points were tracked using GPS receivers with a sampling rate of ≈ 30s and a positioning system error of 10-20m in normal conditions over a period 5 weeks. As deeply discussed in [7], this sample of data indicates strong evidence to the validity and coherence of GPS data and its representativeness power. We have concentrated on the points observed during a one week period from June 14, 2010 to June 20, 2010, consiting of ≈ 4,000 cars per day. To avoid the extraction of stopping T-Flocks (e.g. groups of cars parked in the same area), the traces of the vehicles have been preprocessed by cutting each trajectory in smaller trips: a stop longer than two hours determines the end of the current trip and the start of a new one.

Flock Discovery. According with the T-Flock definition, the traces of the vehicles have been divided into line segments with a time interval of $30s$. The T-Flock patterns are extracted considering a disk radius of $200m$, at least 6 *line segments* –i.e. vehicles moving together for at least 180 seconds– and a minimum flock support of at least 3 vehicles.The discovered flocks on the June 14 data are shown in Figure 1.

Traffic jam detection. To distinguish the traffic jams from all the flocking vehicles, the analyst can apply a set of constraints based on the T-Flock attributes: *duration*, *length*,

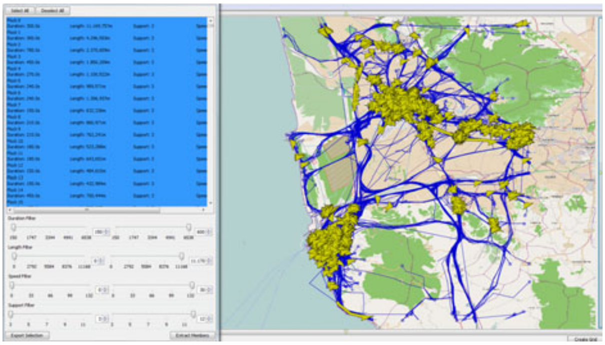


Fig. 1. The complete set of flocks extracted in the Pisa and surrounding area

speed, and *support*. In particular, the flock *speed* constraint can be used to select all the patterns within the given threshold. Figure 2 shows the whole set of flocks discovered, and how it is possible, using the interactive constrains, filter the flocks taking only the slow ones.

Dynamic Speed Constraint. A different approach to detect traffic jams from the set of discovered flocks consists of computing the ration of the speed of a flock with the average speed of all the cars passing through the same area. Actual traffic jams can be identified through their low speed ratio, which means that the involved cars have a considerably lower speed compared to the average speed of passing cars in the considered area at a specific period of time. Figure 3 shows how previously detected traffic jams in Pisa urban area can further be classified with respect to the dynamic speed constraint. It is important to notice that this new constraint is useful to detect events during the day which do not follow the typical mobility behavior in that area. The system is also open to consider different ways to compute the typical speed, which can be for example the using of a map matching algorithm between the flocks and the road network.

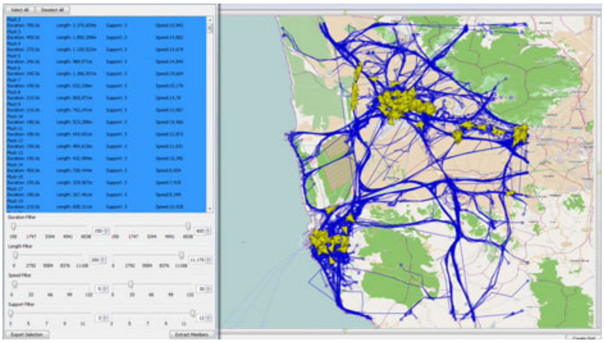


Fig. 2. Detecting traffic jams candidates using the static speed constraint: greater then 5km/h and less than 30 km/h

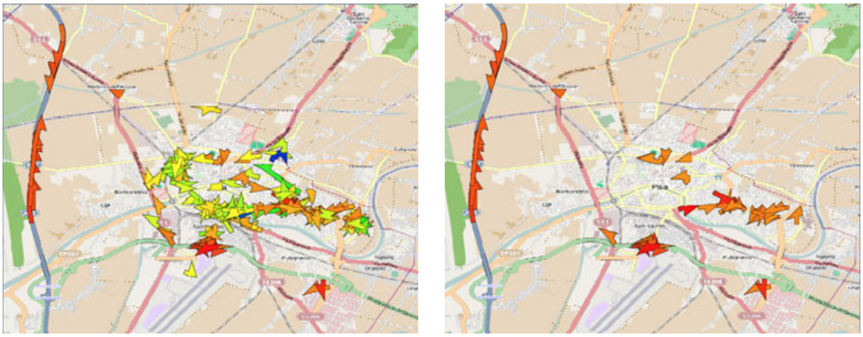


Fig. 3. Applying the dynamic speed constraint on selected candidates highlights how this constraint distinguishes between real traffic jams (red/yellow gradient) and usual congested areas (blue/green gradient). (Left) All the selected flocks in Pisa urban area. without applying any threshold. (Right) Applying a threshold corresponding to a decrement of 60% of average speed.

4 Additional Analyses

During the demo, the case study presented in the previous section will be extended with further analyses: (i) we will show how supplementary information (e.g. points of interest, city gates, meteorological conditions, etc.) can be exploited to add semantics to the models, and (ii) how the traffic jams can be temporally characterized.

References

1. Gudmundsson, J., van Kreveld, M.J.: Computing longest duration flocks in trajectory data. In: GIS (2006)
2. Laube, P., Imfeld, S., Weibel, R.: Discovering relative motion patterns in groups of moving point objects. International Journal of Geographical Information Science 19(6), 639–668 (2005)
3. Gennadu, A., Natalia, A.: A general framework for using aggregation in visual exploration of movement data. The Cartographic Journal 47(1), 22–40 (2010)
4. Trasarti, R., Rinzivillo, S., Pinelli, F., Nanni, M., Monreale, A., Renso, C., Pedreschi, D., Giannotti, F.: Exploring Real Mobility Data with M-Atlas. ECML/PKDD (3), 624–627 (2010)
5. Wachowicz, M., Ong, R., Renso, C., Nanni, M.: Discovering Moving Flock Patterns among Pedestrians through Spatio-Temporal Coherence. International Journal of Geographical Information Science (in press)
6. Benkert, M., Gudmundsson, J., Hübner, F., Wolle, T.: Reporting Flock Patterns. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 660–671. Springer, Heidelberg (2006)
7. Trasarti, R., Giannotti, F., Nanni, M., Pedreschi, D., Renso, C.: A query language for mobility data mining. International Journal of Data Warehousing and Mining (IJDWM) 7(1), 24–45 (2011)

Exploring City Structure from Georeferenced Photos Using Graph Centrality Measures

Katerina Vrotsou[1], Natalia Andrienko[1], Gennady Andrienko[1],
and Piotr Jankowski[2]

[1] Fraunhofer Institute IAIS, Schloss Birlinghoven, 53757 Sankt Augustin, Germany
[2] San Diego State University, 5500 Campanile Drive, San Diego CA 92182-4493, USA

Abstract. We explore the potential of applying graph theory measures of centrality to the network of movements extracted from sequences of georeferenced photo captures in order to identify interesting places and explore city structure. We adopt a systematic procedure composed of a series of stages involving the combination of computational methods and interactive visual analytics techniques. The approach is demonstrated using a collection of Flickr photos from the Seattle metropolitan area.

1 Introduction

Photo sharing websites, such as Panoramio (www.panoramio.com) and Flickr (www.flickr.com), are a popular means of communicating tourist experiences. The posted photographs are freely available, composing a large, rich data source with broad analysis opportunities. Furthermore, they are associated with geographical coordinates and capturing times (geo- and time-referenced), making it possible to regard users' postings as sequences of visited places over time and treat them as trajectories describing their movement. The vast size and multifaceted nature of such data makes their analysis both challenging and interesting.

We explore the potential of applying graph theory measures of centrality to the network of movements extracted from georeferenced photo sequences in order to identify interesting places and explore city structure. To accomplish this we have defined a procedure composed of a series of stages and involving the combination of computational and interactive visual analytics techniques, including clustering, graph analysis, and aggregation.

The analysis of people's activities and movement using georeferenced photographs has been the subject of previous research [5,4,7,6]. The combination of several algorithmic and visual techniques into a concretely defined procedure is what makes this approach unique. The entire procedure is incorporated into a powerful visual analytics framework which allows a user to interactively investigate the results, alter parameters, re-run the process and make refinements.

2 Visual Analytics Approach

The approach we propose involves the retrieval of a graph of movements from a collection of georeferenced photographs captured across a city, and the application of graph theory methods in order to analyse the structural connectivity

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 654–657, 2011.

of this graph and extract information about the structure of the city itself. The procedure we have defined for accomplishing this is composed of four stages, each of which involves several steps performed systematically:

1. *Pre-processing.* Trajectories are extracted from georeferenced photos.
2. *Aggregation of trajectories.* The extracted trajectories are aggregated into moves between generalized places.
3. *Graph of moves and place centrality.* The graph of aggregated moves is considered and its connectivity is assessed by computing centrality scores.
4. *Analysis of results.* The structural connectivity of the graph is interactively analysed using both visual and algorithmic methods.

2.1 Pre-processing

The collection of photographs is first transformed into a set of trajectories. Photographs entered to a photo sharing website by the same person are ordered chronologically forming a sequence of photographed places representing the trajectory of this person in geographical space. Entries that are not properly georeferenced or do not include a time-stamp, and single entry sequences are ignored. Trajectory analysis methods can also be applied for separating trajectories of tourists and locals in order to reveal different characteristics of a city.

2.2 Aggregation of Trajectories

The extracted trajectories are aggregated into flows between generalized places using the method presented in [2] which is performed through the following steps:

1. Characteristic points are extracted from the trajectories.
2. Spatially-bounded density based clustering is applied to group the points by their spatial proximity.
3. The centroids of the retrieved clusters are used as seeds for generating Voronoi polygons.
4. The polygons define the set of *places* that the explored area is divided into and reflect the spatial density of the trajectories.
5. The trajectories are aggregated into *moves* between pairs of *places* by defining transitions between them, and counting the number of transitions present.

Clustering parameters, such as the minimum number of elements and the desired radius of spatial clusters, can be interactively adjusted and thus influence the size and number of extracted places.

2.3 Graph of Moves and Place Centrality

The aggregation results of the photographers' trajectories can be represented using a directed, weighted graph, $G = (V, E)$, where vertex set, V, is the set of extracted places, and edge set, E, is the set of directed edges (i, j) corresponding to moves from place i to place j, each one weighted by the number of moves present. The structural importance of the vertices in the graph, and hence of the extracted places, can be studied by examining their centrality. Several measures of centrality exist for describing different aspects of this structure. We compute and use the following measures (refer to [3] for details):

1. *Degree centrality* of a vertex is defined by the number of its connecting edges. Given our directed, weighted graph we consider four measures of degree centrality: (1) *in-* and (2) *out-degree* are the number of edges connecting into and out of a vertex respectively, and (3) *weighted in-* and (4) *weighted out-degree* are the sum of weights connecting into and out of a vertex.
2. *Closeness centrality* is based on the distance, measured as the size of the shortest path, of a vertex to all other vertices. Closeness centrality of vertex v is defined as the inverse of the total distance of v to all other vertices.
3. *Betweenness centrality* measures centrality by considering the number of shortest paths that pass through a vertex, v, and is defined as the proportion of shortest paths between vertices u and s that pass through v.
4. The *clustering coefficient* represents the likeliness that two neighbours of a vertex v are connected and measures in a sense the importance of a vertex in its immediate neighbourhood. The clustering coefficient of vertex v is defined as the fraction of v's neighbours that are also neighbours of each other.

2.4 Analysis of Results

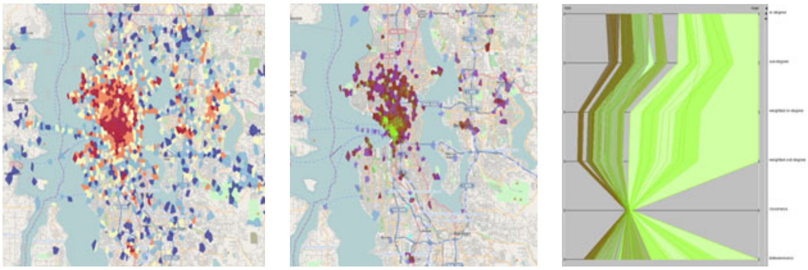
The analysis of the computed centrality is performed through a number of interactive visual analytics tools [1]. Centrality scores are explored, separately or simultaneously, using different visual representations:

- Classified choropleth maps are used to display the scores separately.
- Diagrams are overlaid on a map to make relative comparisons between the centrality scores of places.
- Parallel coordinate plots are used to explore and compare scores of places.
- Clustering of places with respect to their scores is performed to study the spatial distribution of nodes with similar characteristics.
- Dynamic filtering of *places* and *moves* based on centrality is applied to discover interesting places on the map.
- Time graphs and animated maps are used to display time intervals of arbitrary length in order to explore the behaviour of the graph over time.

3 Demonstration: Seattle Metropolitan Area Photos

We apply the presented visual analytics procedure to a collection of Flickr photos from the Puget Sound metropolitan area, Washington State, USA. The data were pre-processed and a dataset was retrieved of 577,053 photos captured by 9,324 photographers, between January 1, 2005 and August 31, 2009.

During the aggregation stage a radius of 500 meters was used for clustering the photos by spatial proximity resulting in the extraction of 2,899 places. Places visited less than 5 times were not considered, resulting in a graph consisting of 1,446 vertices (places) and 40,627 edges, each of them weighted by the number



(a) Closeness centrality (b) Clustering of scores (c) Parallel coordinates

Fig. 1. Representation examples from the exploratory analysis of the Seattle metropolitan area. (a) Closeness centrality in a classified choropleth map. (b) Results of clustering places with respect to centrality scores in a choropleth map. (c) Parallel coordinates plot of centrality scores of four clusters coloured by their corresponding cluster colour.

moves between connecting places. Centrality scores were computed for each of the places in the graph.

Several interesting observations were made during the exploratory analysis of this Seattle dataset, these will be demonstrated in their entirety at the conference (a sample is available at `http://geoanalytics.net/and/slides/pkdd11.pdf`). Examples include the interesting picture of the core and periphery of well connected locations revealed by the high closeness values (fig. 1(a)), the identification of the most well connected places through clustering with respect to centrality scores (fig. 1(b)), and the exploration of the centrality score values themselves using different representations and filtering tools (fig. 1(c)).

References

1. Andrienko, G., Andrienko, N., Bak, P., Keim, D., Kisilevich, S., Wrobel, S.: A conceptual framework and taxonomy of techniques for analyzing movement. Journal of Visual Languages & Computing 22(3), 213–232 (2011)
2. Andrienko, N., Andrienko, G.: Spatial generalization and aggregation of massive movement data. IEEE Transactions on Visualization and Computer Graphics 17(2), 205–219 (2011)
3. Brandes, U., Erlebach, T. (eds.): Network Analysis. LNCS, vol. 3418. Springer, Heidelberg (2005)
4. Crandall, D.J., Backstrom, L., Huttenlocher, D., Kleinberg, J.: Mapping the world's photos. In: 18th International Conference on World Wide Web, pp. 761–770 (2009)
5. Girardin, F., Calabrese, F., Fiore, F.D., Ratti, C., Blat, J.: Digital Footprinting: Uncovering Tourists with User-Generated Content. IEEE Pervasive Computing 7(4), 36–43 (2008)
6. Jankowski, P., Andrienko, N., Andrienko, G., Kisilevich, S.: Discovering Landmark Preferences and Movement Patterns from Photo Postings. Transactions in GIS 14(6), 833–852 (2010)
7. Kisilevich, S., Krstajic, M., Keim, D., Andrienko, N., Andrienko, G.: Event-Based Analysis of People's Activities and Behavior Using Flickr and Panoramio Geotagged Photo Collections. In: 14th Int'l Conf. Information Visualisation, pp. 289–296 (2010)

Author Index